Laboratório Prático do MSDN

Linha de construção de aplicativos empresariais com Windows Presentation Foundation

Ruihua Jin
Ronnie Saurenmann
(ronnies@microsoft.com)
Suíça Microsoft

Traduzido por
Murilo Junot
(murilo.teixeira@makesys.com.br)
Web Designer - Makesys

Versão do original: 1.0
Última atualização: Maio 2007
Versão da tradução: 1.0
Última atualização: Abril 2008

Índice

Sobre este Laboratório Prático – LP (Hands-On Lab) 5
Sobre o que você aprenderá 5
Organização deste Laboratório Prático 6
Descritivo dos Tópicos Usados Nesta Documentação 6
Onde começar 6
Construindo uma réplica da interface do usuário do MS Outlook 2007 (Básico) 7
Tarefa 1: Abrir o projeto no Expression Blend 8
Tarefa 2: Familiarizar com o Expression Blend 9
Tarefa 3: Adicionar um controle DockPanel no LayoutRoot 12
Tarefa 4: Adicionar um ToolBarTray, um StatusBar e um Grid no [DockPanel] 18
Tarefa 5: Dividir o [Grid] em Colunas e Linhas & Adicionar o GridSplitter no [Grid] 24
Tarefa 6: Adicionar um StackPanel e seus controles de herança à célula inferior esquerda do [Grid] 30
Tarefa 7: Adicionar o MyFoldersExpandersControl ao Upper Left Cell do [Grid] 41
Tarefa 8: Adicionar um ToggleButton para Mostrar/Ocultar o Sidebar 42
Tarefa 9: Adicionar um MyInboxExpanderControl à Segunda Coluna do [Grid] 57
Tarefa 10: Adicionar um controle ListView no Display Mail 58
Tarefa 11: Adicionar controles à terceira coluna do [Grid] no conteúdo Display Mail 64
Criando o Sidebar (avançado) 67
Tarefa 1: Criar o Botão Sidebar 67
Tarefa 2: Editar um Modelo de Controle para os Botões Sidebar 70
Tarefa 3: Escrever Manipuladores de Eventos para Mostrar/Ocultar o Painel de Navegação 74
Personalizando o Controle [ListView] (avançado) 76
Tarefa 1: Editar o estilo dos cabeçalhos de colunas 77
Tarefa 2: Editar o ItemContainerStyle do [ListView] 83
Tarefa 3: Adicionar Colunas de Imagem no Importance, Read, Attachment do [ListView] 85
Tarefa 4: Classificar a lista 88
Mais em Expression Blend e Windows Presentation Foundation 90

Sobre este Laboratório Prático – LP (Hands-On Lab)

Sobre o que você aprenderá

Windows Presentation Foundation (WPF) fornece aos desenvolvedores um modelo de programação unificado para construir ricas janelas (Windows) de experiências inteligentes para usuários que incorporam UI (User Interface – Interface com o Usuário), mídias e documentos. Neste Laboratório Prático você aprenderá sobre a construção de um aplicativo de WPF usando o Microsoft Expression Blend. O projeto finalizado, como mostra a figura, é uma réplica do Microsoft Outlook 2007 UI:

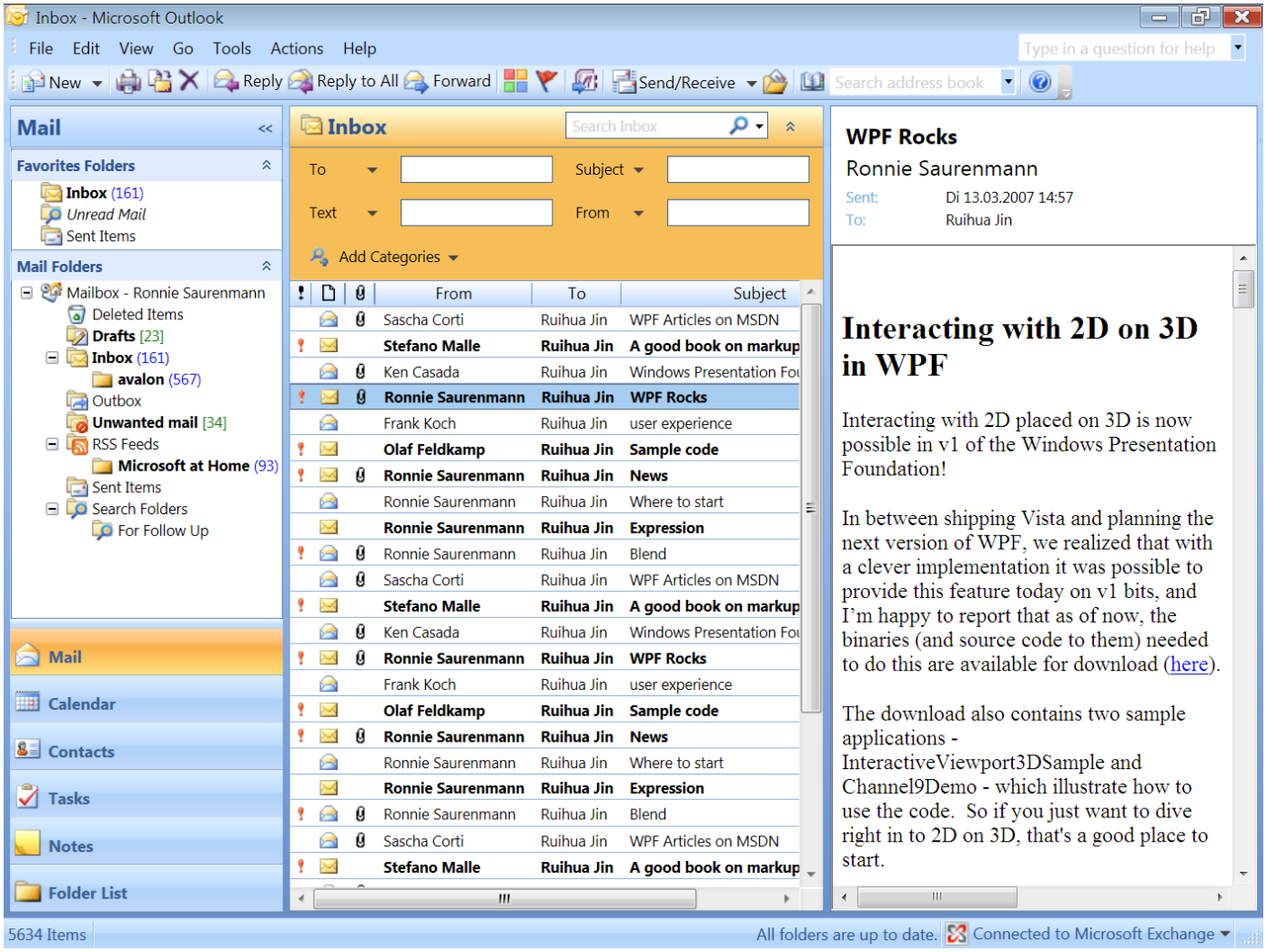

Inbox - Microsoft Outlook
File Edit View Go Tools Actions Help
New Reply Reply to All Forward Send/Receive
Mail
Favorites Folders
Inbox (161)
Unread Mail
Sent Items
Mail Folders
Mailbox - Ronnie Saurenmann
Deleted Items
Drafts [23]
Inbox (161)
avalon (567)
Outbox
Unwanted mail [34]
RSS Feeds
Microsoft at Home (93)
Sent Items
Search Folders
For Follow Up
Mail
Calendar
Contacts
Tasks
Notes
Folder List
Inbox
Search Inbox
To
Subject
Text
From
Add Categories
From To Subject
Sascha Corti Ruihua Jin WPF Articles on MSDN
Stefano Malle Ruihua Jin A good book on markup
Ken Casada Ruihua Jin Windows Presentation Fou
Ronnie Saurenmann Ruihua Jin WPF Rocks
Frank Koch Ruihua Jin user experience
Olaf Feldkamp Ruihua Jin Sample code
Ronnie Saurenmann Ruihua Jin News
Ronnie Saurenmann Ruihua Jin Where to start
Ronnie Saurenmann Ruihua Jin Expression
Ronnie Saurenmann Ruihua Jin Blend
Sascha Corti Ruihua Jin WPF Articles on MSDN
Stefano Malle Ruihua Jin A good book on markup
Ken Casada Ruihua Jin Windows Presentation Fou
Ronnie Saurenmann Ruihua Jin WPF Rocks
Frank Koch Ruihua Jin user experience
Olaf Feldkamp Ruihua Jin Sample code
Ronnie Saurenmann Ruihua Jin News
Ronnie Saurenmann Ruihua Jin Where to start
Ronnie Saurenmann Ruihua Jin Expression
Ronnie Saurenmann Ruihua Jin Blend
Sascha Corti Ruihua Jin WPF Articles on MSDN
Stefano Malle Ruihua Jin A good book on markup
WPF Rocks
Ronnie Saurenmann
Sent: Di 13.03.2007 14:57
To: Ruihua Jin
Interacting with 2D on 3D in WPF
Interacting with 2D placed on 3D is now possible in v1 of the Windows Presentation Foundation!
In between shipping Vista and planning the next version of WPF, we realized that with a clever implementation it was possible to provide this feature today on v1 bits, and I'm happy to report that as of now, the binaries (and source code to them) needed to do this are available for download (here).
The download also contains two sample applications - InteractiveViewport3DSample and Channel9Demo - which illustrate how to use the code. So if you just want to dive right in to 2D on 3D, that's a good place to start.
5634 Items
All folders are up to date.
Connected to Microsoft Exchange

Figura 1 O projeto finalizado.

As seguintes características de WPF são envolvidas:

- Controles de layout e design como **Grid**, **DockPanel**, **StackPanel**, etc.
- Controles comuns de interface com o usuário como **Button**, **ToggleButton**, **Border**, **TextBlock**, **ListView**, etc.
- Drawing paths (trajetos de desenho)
- Objetos de transformação
- Solid brushes (pincéis sólidos), linear gradient brushes (pincéis gradientes lineares)
- Data binding (ligação de dados), data converters (conversores de dados)
- Styles (estilos), control templates (modelos de controle), template binding (ligação de templates)
- Data templates (modelos de dados)
- Property triggers (disparadores da propriedade), data triggers (disparadores de dados)
- Event handling (manipulação de eventos)

Organização deste Laboratório Prático

O Laboratório Prático é dividido em três cursos: um curso básico e dois cursos avançados.

Parte I – Componentes de construção do Outlook UI (básico)

A primeira parte do conhecimento básico, do Laboratório Prático de WPF, é dada. Todos os componentes importantes do Outlook UI são construídos durante este curso.

Parte II – Personalizando Buttons (botões) (avançado)

Na segunda parte você aprenderá sobre como personalizar controles **Buttons** do seu próprio gosto.

Parte III – Personalizando ListView (exibição de lista) (avançado)

Na terceira parte você aprenderá sobre como personalizar um controle **ListView** do seu gosto.

Descritivo dos Tópicos Usados Nesta Documentação

Ponto importante para saber do coração do WPF

Dica, Nota

Exercício

Introdução passo a passo em MS Expression Blend

Linhas de código (para economizar muita digitação, nós preparamos linhas de código que podem ser copiadas e então coladas. Os arquivos para as linhas de código estão localizados sob o diretório **project folder\OutlookUI HOL\code snippets**, onde o **project folder** é o diretório em que você copiou o projeto).

Onde começar

É importante para nós que você se familiarize com tantas características de WPF quanto possível durante o Laboratório Prático, assim nós preparamos diversos projetos que você pode começar sob o **project folder\OutlookUI HOL\Initial Projects**, sentindo-se livre para escolher seu próprio ponto de partida. A seguinte tabela lhe dá uma visão geral:

Ponto de partida	**Nome do projeto**
Básico, etapa 1	OutlookUI HOL Basic - Initial
Básico, etapa 6	OutlookUI HOL Basic6 - Initial
Básico, etapa 8	OutlookUI HOL Basic8 - Initial
Básico, etapa 10	OutlookUI HOL Basic10 - Initial
Avançado (Sidebar)	OutlookUI HOL Basic - Final
Avançado (ListView)	OutlookUI HOL Basic - Final

Construindo uma réplica da interface com o usuário do MS Outlook 2007 (Básico)

Neste curso básico, nós construiremos um aplicativo de WPF com todos os componentes importantes que você encontra no MS Outlook 2007. Depois que você terminar o curso básico, seu aplicativo terá a seguinte aparência:

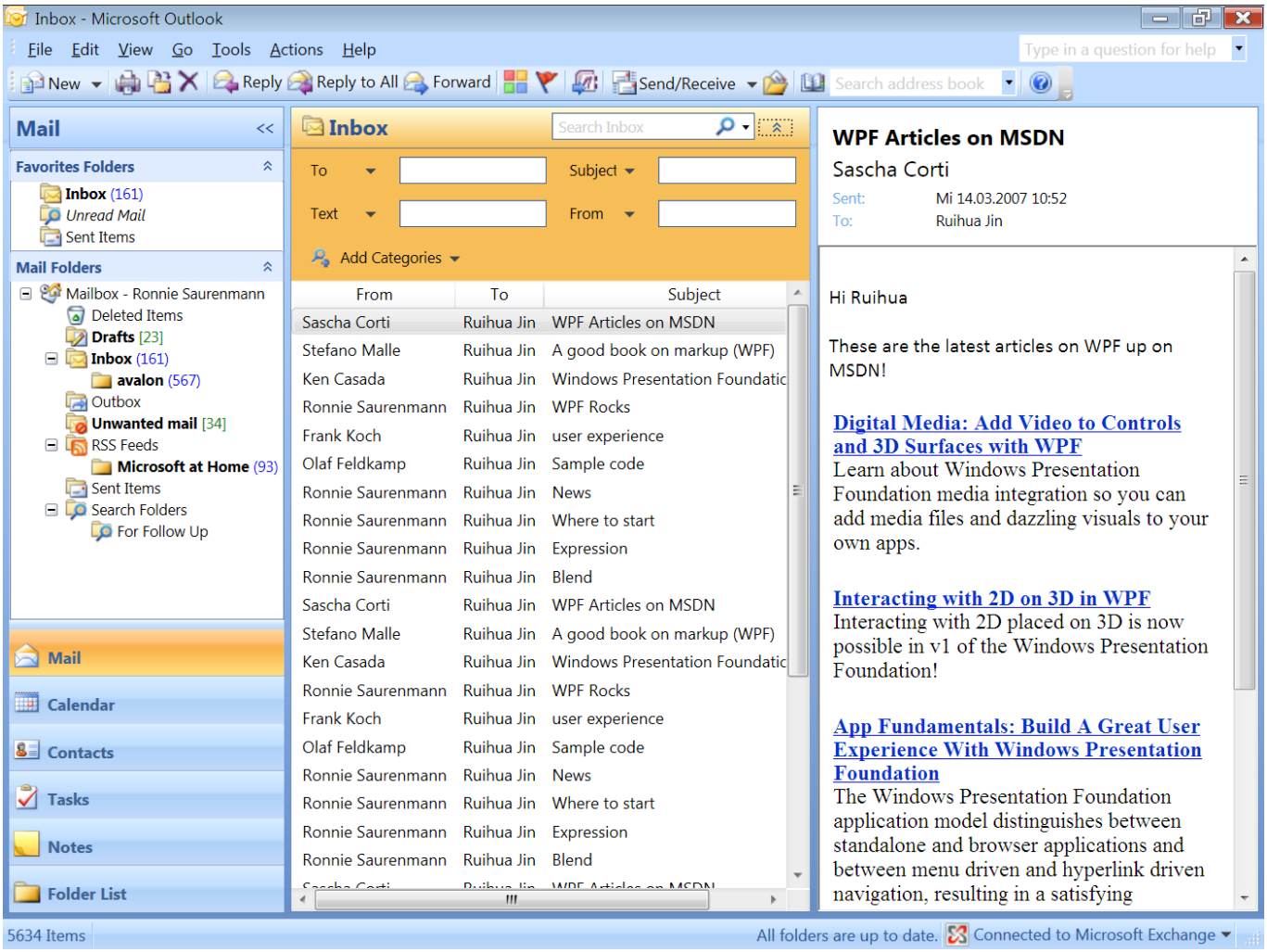

Inbox - Microsoft Outlook
File Edit View Go Tools Actions Help
New Reply Reply to All Forward Send/Receive
Mail
Favorites Folders
Inbox (161)
Unread Mail
Sent Items
Mail Folders
Mailbox - Ronnie Saurenmann
Deleted Items
Drafts [23]
Inbox (161)
avalon (567)
Outbox
Unwanted mail [34]
RSS Feeds
Microsoft at Home (93)
Sent Items
Search Folders
For Follow Up
Mail
Calendar
Contacts
Tasks
Notes
Folder List
Inbox
Search Inbox
To
Subject
Text
From
Add Categories
From To Subject
Sascha Corti Ruihua Jin WPF Articles on MSDN
Stefano Malle Ruihua Jin A good book on markup (WPF)
Ken Casada Ruihua Jin Windows Presentation Foundatic
Ronnie Saurenmann Ruihua Jin WPF Rocks
Frank Koch Ruihua Jin user experience
Olaf Feldkamp Ruihua Jin Sample code
Ronnie Saurenmann Ruihua Jin News
Ronnie Saurenmann Ruihua Jin Where to start
Ronnie Saurenmann Ruihua Jin Expression
Ronnie Saurenmann Ruihua Jin Blend
Sascha Corti Ruihua Jin WPF Articles on MSDN
Stefano Malle Ruihua Jin A good book on markup (WPF)
Ken Casada Ruihua Jin Windows Presentation Foundatic
Ronnie Saurenmann Ruihua Jin WPF Rocks
Frank Koch Ruihua Jin user experience
Olaf Feldkamp Ruihua Jin Sample code
Ronnie Saurenmann Ruihua Jin News
Ronnie Saurenmann Ruihua Jin Where to start
Ronnie Saurenmann Ruihua Jin Expression
Ronnie Saurenmann Ruihua Jin Blend
WPF Articles on MSDN
Sascha Corti
Sent: Mi 14.03.2007 10:52
To: Ruihua Jin
Hi Ruihua
These are the latest articles on WPF up on MSDN!
Digital Media: Add Video to Controls and 3D Surfaces with WPF
Learn about Windows Presentation Foundation media integration so you can add media files and dazzling visuals to your own apps.
Interacting with 2D on 3D in WPF
Interacting with 2D placed on 3D is now possible in v1 of the Windows Presentation Foundation!
App Fundamentals: Build A Great User Experience With Windows Presentation Foundation
The Windows Presentation Foundation application model distinguishes between standalone and browser applications and between menu driven and hyperlink driven navigation, resulting in a satisfying
5634 Items
All folders are up to date.
Connected to Microsoft Exchange

Figura 2 O projeto do curso básico finalizado.

Você aprenderá sobre as seguintes características neste curso básico:

- Interface com o usuário do Expression Blend
- Controles de layout e design como **Grid**, **DockPanel**, **StackPanel**, etc.
- Controles comuns de interface com o usuário como **Button**, **ToggleButton**, **Border**, **TextBlock**, **ListView**, etc.
- Solid brushes, linear gradient brushes
- Drawing paths
- Objetos de transformação
- Data binding da propriedade de um elemento à propriedade de outro elemento
- Data binding do contexto dos dados de um elemento a uma origem de dados, e a uns dados de XML que ligam a propriedade de um elemento a um elemento de XML
- Data converters
- Styles
- Control Templates
- Property triggers
- Event handling

Tarefa 1: Abrir o projeto no Expression Blend

O seguinte procedimento descreve como abrir o projeto **OutlookUI HOL Basic – Initial**:

1. No Expression Blend, clique em **File**, e clique então em **Open Project…**. A caixa de diálogo **Open Project** é aberta.
2. Vá para o diretório **project folder\OutlookUI HOL\Initial Projects\OutlookUI HOL Basic – Initial**, onde o *project folder* é a pasta em que você copiou o projeto. Selecione **OutlookUI HOL.sln**, e clique então em **Open**.

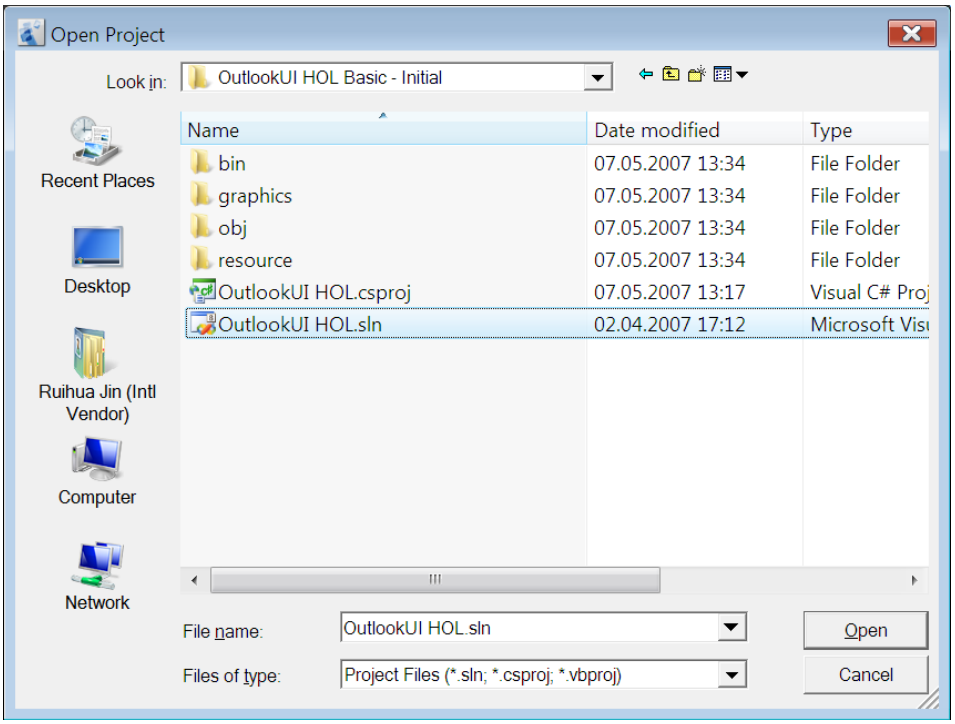


Figura 3 A caixa de diálogo Open Project para abrir o projeto OutlookUI HOL Basic - Initial.

Agora você tem um projeto carregado na memória e vamos começar a construir uma interface com o usuário impressionante.

Tarefa 2: Familiarizar com o Expression Blend

A interface com o usuário do Expression Blend é dividida em 6 áreas principais:

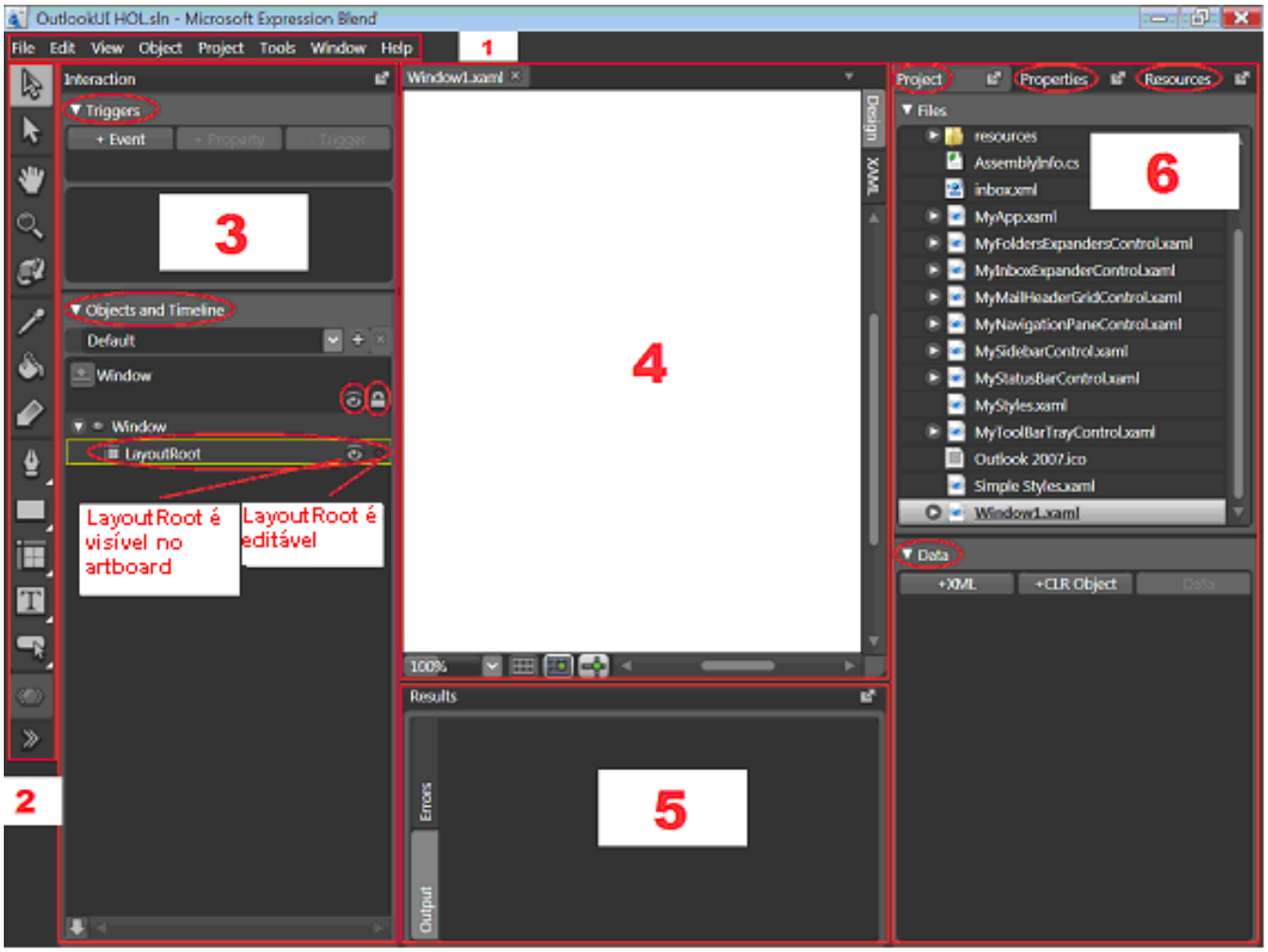


Figura 4 A interface com o usuário padrão do Expression Blend.

1. O **Menu**.
2. O **ToolBox** (caixa de ferramentas). O **ToolBox** fornece vários recursos que você pode desenhar no artboard. Pare o mouse sobre as ferramentas para ver seus nomes. Algumas das ferramentas incluem um triângulo pequeno que você pode clicar com o botão direito para abrir uma lista de outros recursos disponíveis. Para ter uma lista completa dos recursos clique no botão **Asset Library** (Biblioteca do recurso) .
3. O painel **Interaction** (painel de Interação).
    - A seção **Triggers** (disparadores). Sob **Triggers**, no painel **Interaction**, você pode definir o disparador da propriedade ou o disparador do evento, para algum controle.
    - A seção **Objects and Timeline** (objetos e linha do tempo). Se você tem um arquivo XAML como **Window1.xaml** aberto no artboard no modo **Design**, a estrutura hierárquica do documento XAML é indicada sob **Objects and Timeline**. Uma seta é mostrada ao lado de um elemento, se este elemento tem elementos de herança clique na seta para expandir ou esconder o elemento. Para um aplicativo de WPF, o elemento da raiz é **Window** ou **Page**, e você pode desenhar vários controles de sua própria escolha sob o elemento **Window** ou **Page**.

Durante este LP você pode encontrar o seguinte problema: o artboard de repente se torna branco e você acredita que perdeu todo o seu trabalho. Não entre em pânico, apenas certifique-se de que esteja indicado ao lado de todos os seus controles, na seção **Objects and Timeline**, como está indicado, ao lado de **LayoutRoot**, na figura 4. O ícone indica que os controles relacionados e seus elementos de herança são visíveis no artboard. Se você vê em vez de , é provável que você tenha clicado em inconscientemente; apenas clique em para ter seus controles de volta no artboard.

Similarmente, o ícone diz-nos se os controles relacionados e seus elementos de herança são editáveis. representa “bloqueado”, e representa “editável”, que é o padrão. Lembre-se sempre de verificar o cadeado de um controle quando está querendo saber por que você não pode selecionar ou editar controle.

4. O **artboard** (placa de arte). O **artboard** oferece dois modos de exibição: o modo **Design** e o modo **XAML**. O modo **Design** mostra a aparência do seu aplicativo. O modo **XAML** mostra todo o documento XAML. Clique na guia **Design** e na guia **XAML** para alternar entre esses dois modos de exibição.

Expression Blend, XAML, Windows Presentation Foundation e .NET Framework 3.0

Microsoft® Expression Blend™ é uma novidade, com ferramentas completas para profissionais de design fazerem criações atraentes, sofisticadas e interface com o usuário com conexão Web para aplicativos baseados no Windows®.
Extensible Application Markup Language, ou **XAML** (em inglês, a pronuncia é "zammel"), é uma linguagem de marcação com base em XML desenvolvida pela Microsoft. XAML é a linguagem por trás da apresentação visual de um aplicativo que você desenvolve no Microsoft® Expression Blend™, assim como HTML é a linguagem por trás da apresentação visual de uma página da Web. Criar um aplicativo no Expression Blend significa escrever código XAML à mão ou visualmente trabalhando no modo **Design** do Expression Blend.
XAML é parte do **Microsoft® Windows Presentation Foundation (WPF)**. O WPF está na categoria de recursos no **Microsoft® .NET Framework 3.0** que lidam com a apresentação visual dos aplicativos baseados no Windows e aplicativos de cliente baseado em navegador Web.
O **XAML**, para qualquer documento determinado no Expression Blend, é armazenado em um arquivo **.xaml**. Se há código subjacente para seu documento XAML, esse código é armazenado em um arquivo de mesmo nome, com a extensão adicional de **.cs** ou **.vb**. Por exemplo, se seu documento XAML for chamado **Window1.xaml**, o código por trás do arquivo vai se chamar **Window1.xaml.cs**, se a linguagem de programação for em código C#.

Para obter a exibição completa do **artboard** escolha **Hide Panels** (ocultar painéis) (**F4** ou **TAB**) no menu **Window**.

5. O painel **Results** (painel de Resultados). O painel **Results** exibe informações como avisos, erros que se originam de um documento XAML errado ou informações de saída durante uma criação de projeto.

Analisador WPF (WPF parser)

Quando você cria seu projeto, o analisador WPF lê o arquivo **.xaml** para o projeto e relata quaisquer erros resultantes. Da mesma forma, quando você abre um projeto existente no Expression Blend, o analisador lê os arquivos **.xaml** que são incluídos na sua pasta de projeto e tenta analisar os elementos e exibir os documentos no artboard no modo **Design**. Em ambos os casos, se o analisador encontrar erros, o artboard é desativado e o Expression Blend exibe uma mensagem de erro com um link para abrir o modo de exibição XAML para que você possa resolver os erros. As análises de erros também são relatadas na guia **Errors** (erros) no painel **Results**.

6. Os painéis **Project**/**Properties**/**Resources** (painéis de Projeto/Propriedades/Recursos).
    - O painel **Project** (painel do projeto).
        - A seção **Files** (arquivos). Você obtém uma lista de todos os arquivos envolvidos no projeto atual em **Files**, no painel **Project**. Esse é o local onde você abre arquivos, adiciona arquivos, conjunto de módulos (assemblies) ou outros recursos relacionados, como imagens, etc., ao seu projeto.

Alterne para o painel **Project** e dê uma olhada na estrutura de arquivos do nosso projeto.

        - A seção **Data** (de dados). Em **Data** no painel **Project** você pode adicionar fontes de dados de objeto XML ou CLR ao projeto. Você aprenderá adicionar fonte de dados XML e controles de dados vinculados a ela neste laboratório.
    - O painel **Properties** (painel de propriedades). O painel **Properties** mostra todas as propriedades e eventos de um controle selecionado.

Seleção de controle para configurar as propriedades

Para **selecionar** um controle, clique no controle, uma vez, em **Objects and Timeline**, depois disso, o nome do elemento é realçado em cinza no plano de fundo e o elemento correspondente no artboard é realçado por um contorno limitador azul (veja figura 5).

Selecione o elemento **LayoutRoot** e dê uma olhada nas suas propriedades.

Observe que as propriedades são categorizadas por tipo, como **Brushes**, **Appearance**, etc., que permitem que você localize facilmente uma propriedade específica. Se você souber o nome de uma propriedade ou parte dele, você também pode digitá-lo na caixa **Search** para obter somente as propriedades que você deseja. Lembre-se de limpar a caixa **Search** quando você não desejar filtrar mais propriedades.
Para exibir todas as propriedades de uma determinada categoria, clique na seta , se existir.

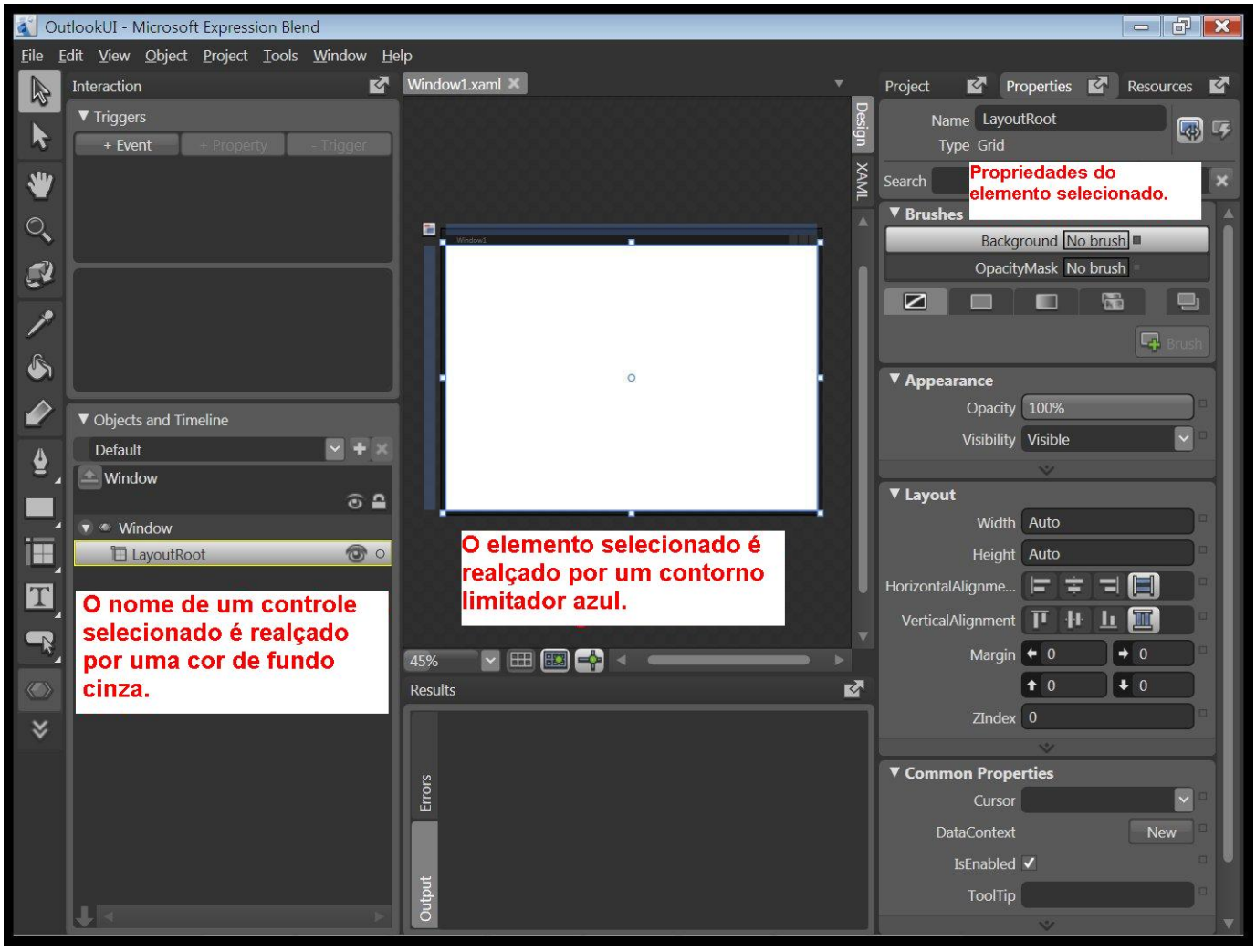


Figura 5 O LayoutRoot é o elemento selecionado.

- O painel **Resources** (painel de recursos). O painel **Resources** mostra todos os recursos definidos para o projeto atual. Exemplos de recursos são brushes (pincéis), styles (estilos) e control templates (modelos de controle).

Alterne para o painel **Resources** e clique em **MyStyles.xaml**, dê uma olhada nos brushes que irão contribuir para o nosso **OutlookUI**.

Tarefa 3: Adicionar um controle DockPanel no LayoutRoot

Antes de começar adicionando controles no seu aplicativo WPF, você deve primeiro tomar uma decisão de qual painel de layout os controles serão inseridos. Há três painéis de layout usados com freqüência: **Grid**, **DockPanel** e **StackPanel**.

- Um controle **Grid** organiza elementos de herança em um layout de linhas e colunas que constituem uma grade. O elemento **LayoutRoot** fornecido pelo Expression Blend por padrão, quando você inicia com um novo projeto, é um **Grid**.
- Um controle **DockPanel** organiza elementos de herança para que eles permaneçam em uma borda do painel: **Left**, **Top**, **Right** ou **Bottom**.
- Um controle **StackPanel** organiza elementos de herança em uma única linha que pode ser orientada horizontalmente ou verticalmente.

Para o nosso projeto escolhemos um controle **DockPanel** como o recipiente raiz. O procedimento a seguir mostra como adicionar um controle **DockPanel** no **LayoutRoot**.

1. Em **Objects and Timeline**, dê um duplo-clique no **LayoutRoot** para ativá-lo.

Ativação de controle para adicionar elementos de herança

Você precisa **ativar** o elemento antes de adicionar elementos de herança a ele. Dê um duplo-clique no elemento pai, em **Objects and Timeline**, no painel **Interaction**, depois disso, será exibida uma caixa amarela delimitadora em torno do nome do elemento, que indica que o elemento está ativo e você pode agora adicionar elementos filho a ele (veja figura 6).

Seleção de controle vs. ativação de controle

Para editar as propriedades de um elemento, você só precisa **selecionar** o elemento: Clique no elemento, uma vez, em **Objects and Timeline**. Você pode selecionar um elemento sem ativá-lo.

Você também pode usar a ferramenta **Selection** , do **Toolbox**, para selecioná-lo e ativar o respectivo elemento com um duplo-clique no artboard.

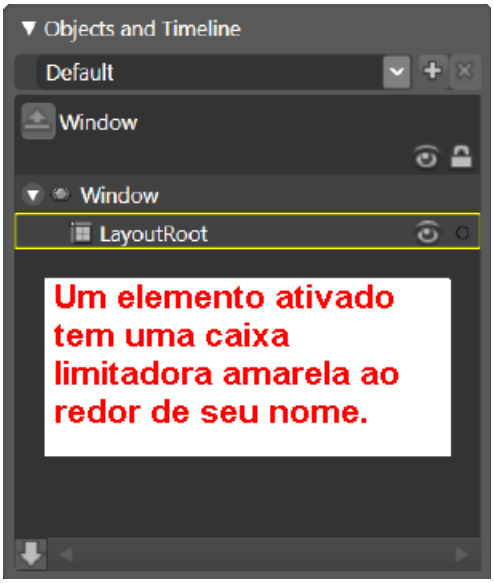


Figura 6 O LayoutRoot é o elemento ativo.

2. Adicione a ferramenta **DockPanel** no **Toolbox**. 7

O ícone indica que há uma introdução detalhada na tela do Expression Blend. O número ao lado dele é o número da figura. Sugerimos que você sempre dê uma olhada, primeiro, na figura, se houver uma.

A ferramenta **DockPanel** não é mostrada no **ToolBox** por padrão, portanto, você deve primeiro adicionar a ferramenta **DockPanel** no **ToolBox**: Clique no botão **Asset Library** , no **ToolBox**, para abrir a caixa de diálogo **Asset Library**, que exibe todas as ferramentas disponíveis para o projeto. Para localizar rapidamente a ferramenta **DockPanel**, digite **“ DockPanel ”** na caixa de texto no canto superior esquerdo, em seguida, clique no **DockPanel** mostrado na caixa de diálogo.

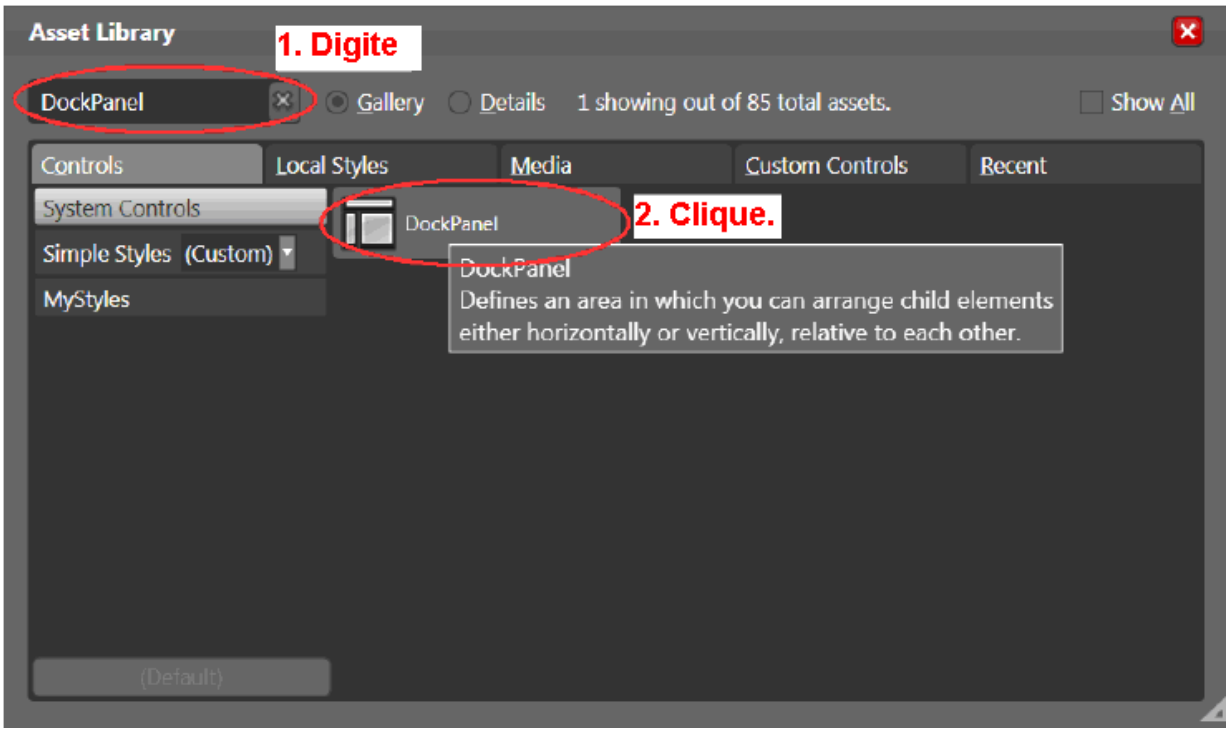


Figura 7 Adicionando a ferramenta DockPanel no ToolBox.

Agora você pode ver o ícone da ferramenta **DockPanel** no **ToolBox**.

3. Adicione um controle **DockPanel** no **LayoutRoot**. 8 9
Há duas formas comuns para adicionar um controle filho a um controle pai, e em ambas requerem que o controle pai seja ativado primeiro. Com o controle pai ativado, execute uma das seguintes operações:
   a. Clique na ferramenta do controle filho, no **ToolBox**, mova o mouse na posição onde deseja que o controle seja localizado, no artboard, e em seguida, desenhe um controle do tamanho desejado enquanto mantém pressionado o botão esquerdo do mouse (veja figura 8).
   b. Dê um duplo-clique na ferramenta do controle filho, no **ToolBox**, depois disso, o controle é adicionado automaticamente ao controle atualmente ativo com seu tamanho padrão (veja figura 9).

Agora deve ser possível adicionar um controle **DockPanel** no **LayoutRoot**.

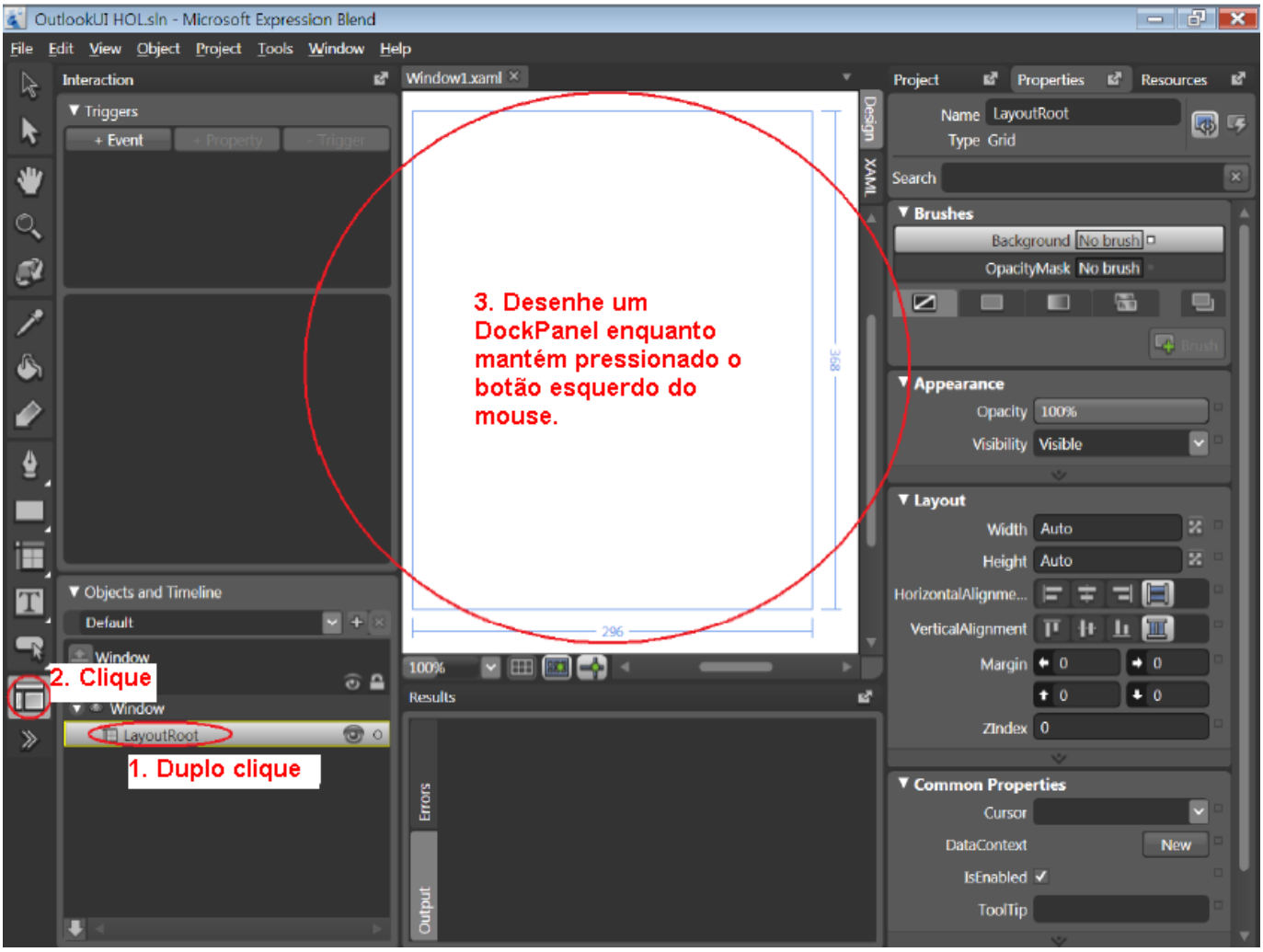


Figura 8 Adicionando um controle DockPanel no LayoutRoot.

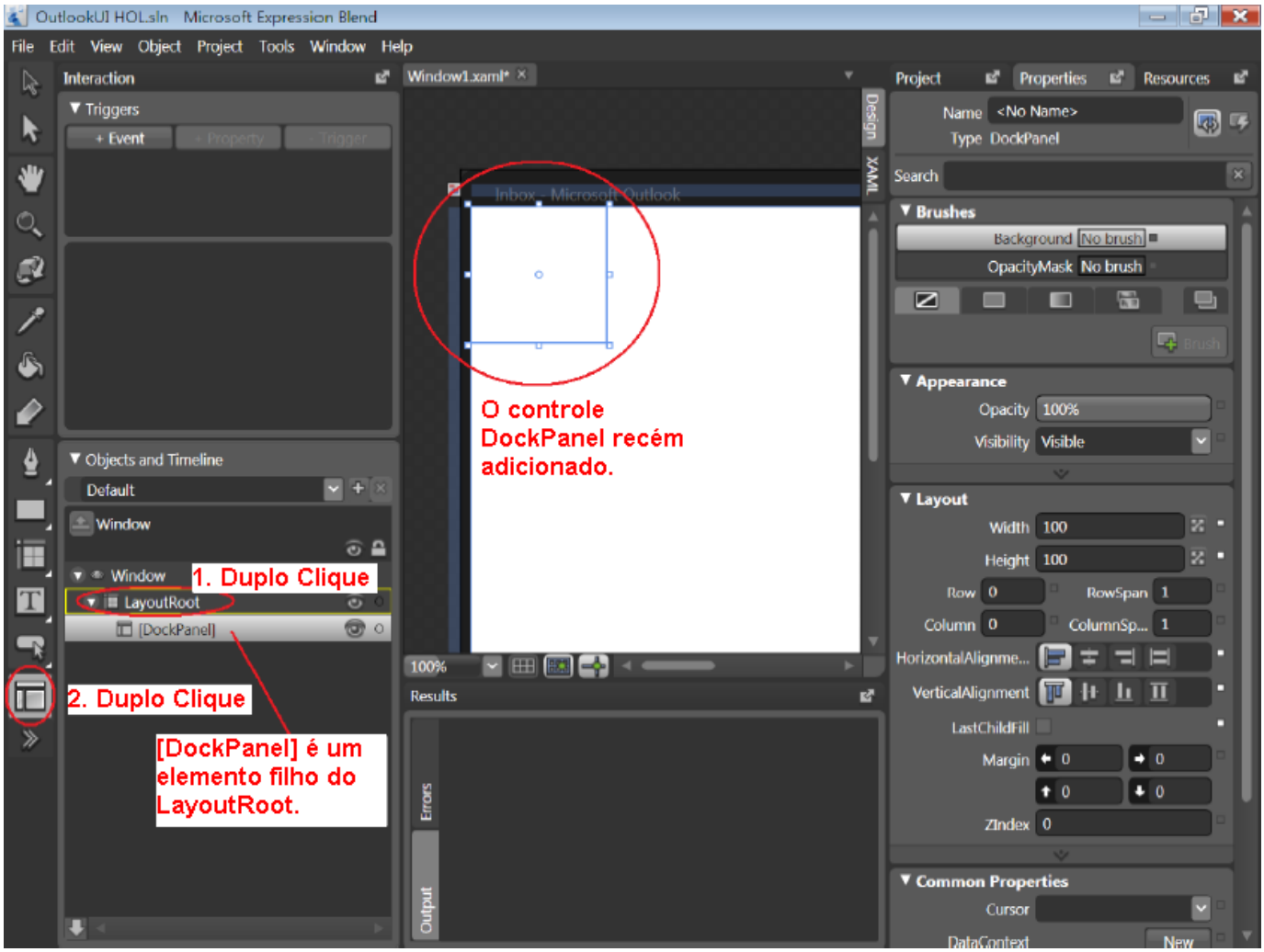


Figura 9 Adicionando um controle DockPanel no LayoutRoot.

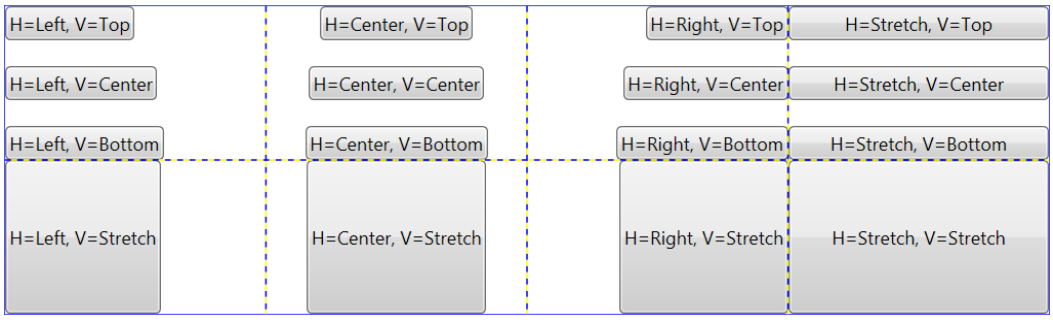
4. Configure as propriedades de layout do **[DockPanel]**. **10**

Alignment, Margin e Padding (alinhamento, margem e preenchimento)

A classe **FrameworkElement** expõe várias propriedades que são usadas para posicionar precisamente elementos de herança. Este tópico aborda quatro das propriedades mais importantes: **HorizontalAlignment**, **VerticalAlignment**, **Margin** e **Padding**.

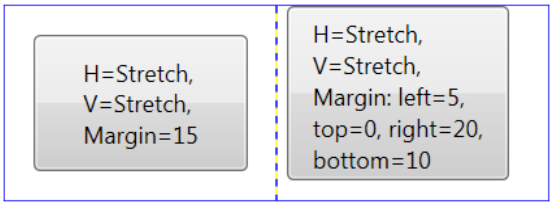
Propriedades do Alignment (alinhamento)

As propriedades **HorizontalAlignment** e **VerticalAlignment** descrevem como um elemento de herança deve ser posicionado dentro de um elemento pai. A propriedade **HorizontalAlignment** declara as características de alinhamento horizontal para aplicar nos elementos de herança. Os valores possíveis da propriedade **HorizontalAlignment** são **Left** (esquerda), **Center** (centro), **Right** (direita) e **Stretch** (estender).
A propriedade **VerticalAlignment** descreve as características de alinhamento vertical para aplicar nos elementos de herança. Os valores possíveis para a propriedade **VerticalAlignment** são **Left** (esquerda), **Center** (centro), **Right** (direita) e **Stretch** (estender).
A figura a seguir mostra diferentes valores de combinações de **HorizontalAlignment** (H) e **VerticalAlignment** (V) do controle **Button** que é elemento filho do controle **Grid**, e cada **Button** é posicionado em uma célula da determinada grade.



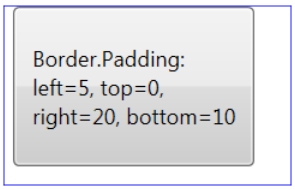
Propriedades de Margin (margem)

A propriedade **Margin** descreve a distância entre um elemento e seus filhos ou pares. O valor de Margin pode ser uniforme, podendo utilizar a sintaxe **Margin = "15"**. Com essa sintaxe, um Margin uniforme de 15 pixels independente de dispositivo poderia ser aplicado ao elemento. Os valores de Margin podem também assumir o formato de quatro valores distintos, cada valor descrevendo uma margem distinta para aplicar à esquerda, superior, direita e inferior (nesta ordem), como **Margin = "0,10,5,25"**.
A figura a seguir mostra dois controles **Button**s que são elementos filho de um controle **Grid**. O **Button** esquerdo tem um valor de Margin uniforme de 15, e o **Button** direito tem quatro valores Margin distintos de **Margin = ” 5, 0, 20, 10”**.



Propriedades de Padding (preenchimento)

A propriedade **Padding** é exposta em apenas algumas classes, principalmente como uma conveniência: **Block**, **Border**, **Control** e **TextBlock** são exemplos de classes que expõem a propriedade **Padding**. A propriedade **Padding** amplia eficazmente o tamanho de um elemento filho pelo valor especificado **Thickness**.
A figura a seguir mostra um controle **Border** que contém um botão. O valor **Padding** do controle **Border** é **“5, 0, 20, 10”**.



Quando desejamos que o **[DockPanel]** seja o recipiente raiz de todos os nossos controles, nós devemos fazê-lo cobrir a área inteira do **LayoutRoot**. Com **[DockPanel]** selecionado, vá para o painel **Properties**, digite **“Auto”** nas caixas de texto **Width** e **Height**, clique nos dois botões **“Stretch”** das propriedades **HorizontalAlignment** e **VerticalAlignment** e defina as margens esquerda, direita, superior e inferior para 0.

O que fizemos aqui foi realmente redefinir essas propriedades para seus valores padrão, para que você possa também clicar no quadrado pequeno ao lado da propriedade para abrir o menu contextual e escolher **Reset**; o valor da propriedade é definido, então, automaticamente, com o valor padrão.

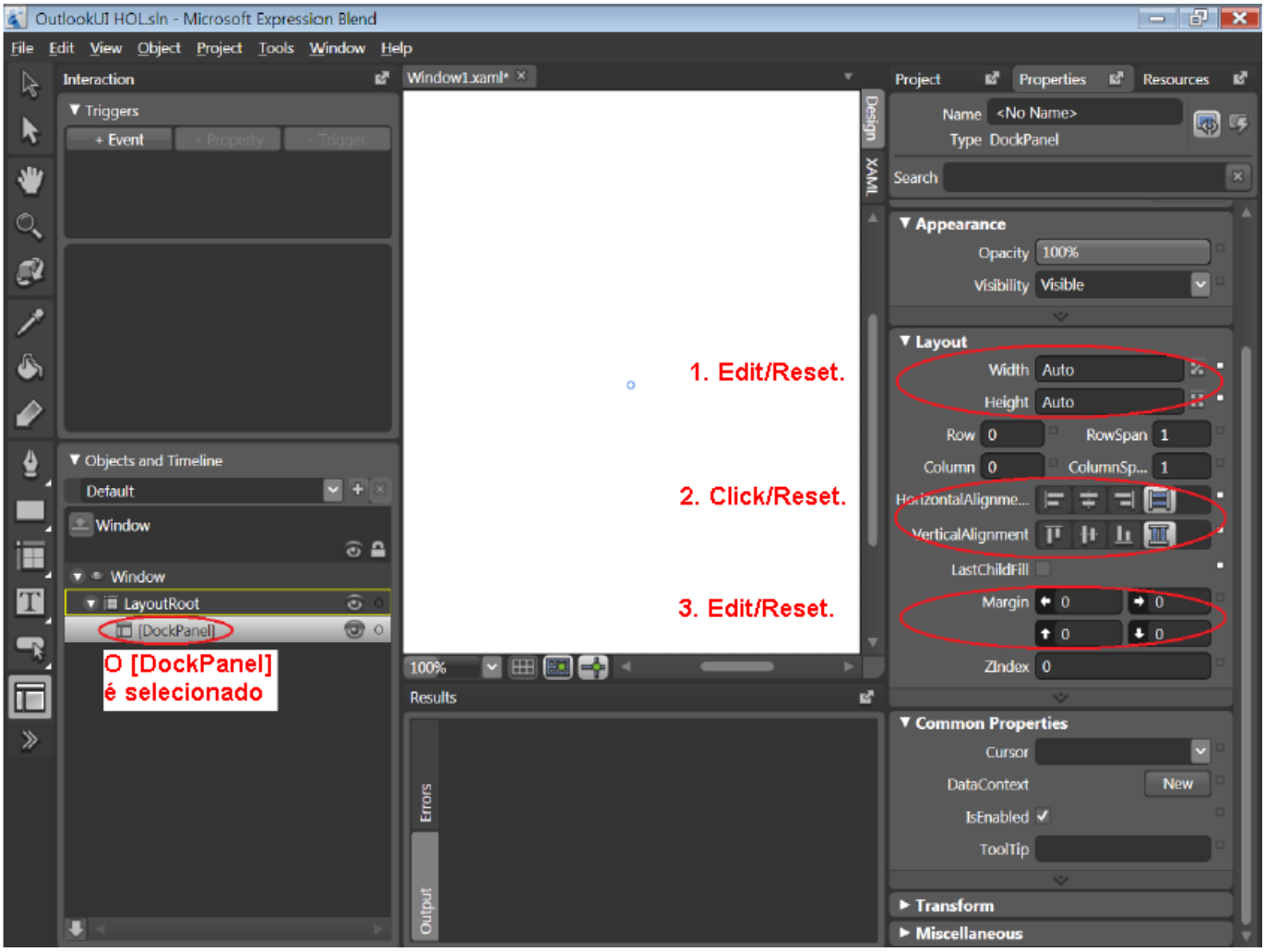


Figura 10 Configurando as propriedades do [DockPanel].

Agora, temos um controle **DockPanel**, e vamos preencher o controle **[DockPanel]** com um **ToolBarTray**, um **StatusBar** e um **Grid**.

Tarefa 4: Adicionar um ToolBarTray, um StatusBar e um Grid no [DockPanel]

Nosso projeto foi fornecido com um controle **ToolBarTray** personalizado e um controle **StatusBar** personalizado, e, ambos são definidos como controles do usuário, chamados **MyToolBarTrayControl** e **MyStatusBarControl**.

1. Adicione um **MyToolBarTrayControl** no **[DockPanel]**.

   1) Adicione a ferramenta **MyToolBarTrayControl** no **ToolBox**. **11**
   Abra a caixa de diálogo **Asset Library**, selecione a guia **Custom Controls**, digite **MyToolBarTrayControl** na caixa de texto e clique em **MyToolBarTrayControl**. Agora você pode ver a ferramenta no **ToolBox**.

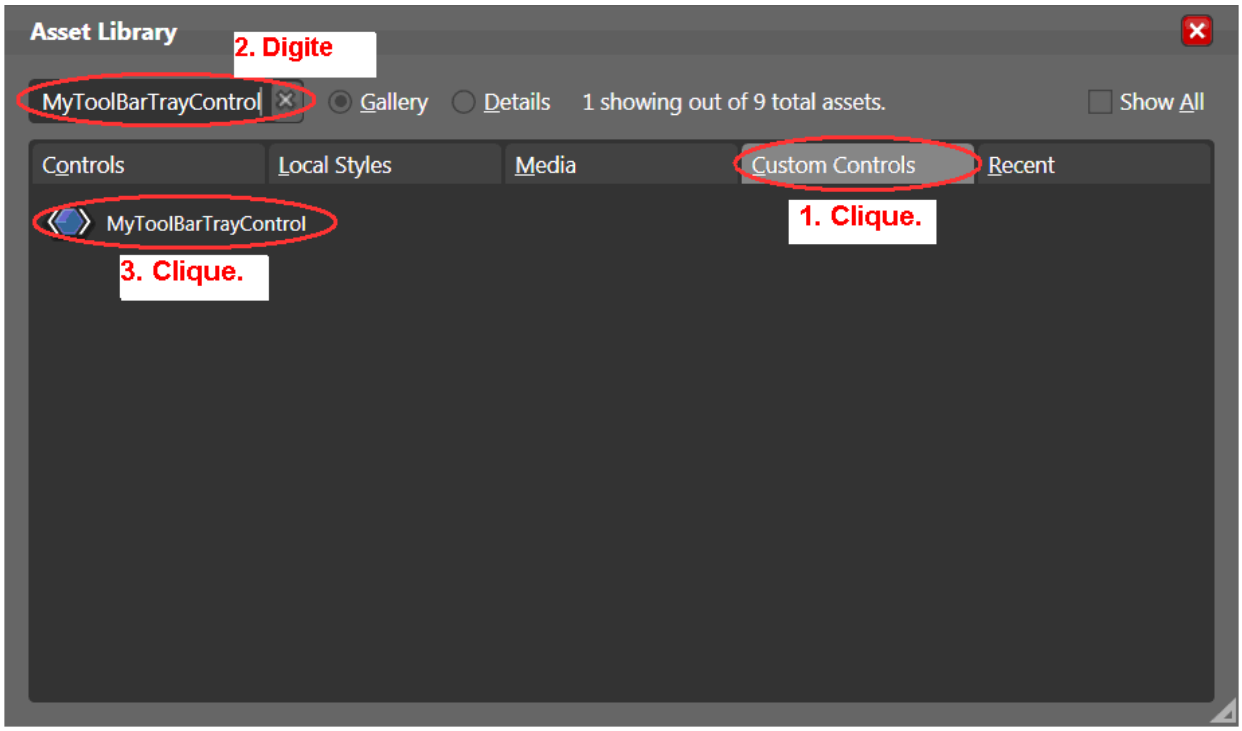


Figura 11 Adicionando a ferramenta MyToolBarTrayControl no ToolBox.

   2) Adicione um **MyToolBarTrayControl** no **[DockPanel]** e defina a propriedade **Dock** como **Top**.
   12
   Ative o **[DockPanel]**, dê um duplo-clique na ferramenta **MyToolBarTrayControl** no **ToolBox**, e defina a propriedade **Dock** do **[MyToolBarTrayControl]** como **Top**.

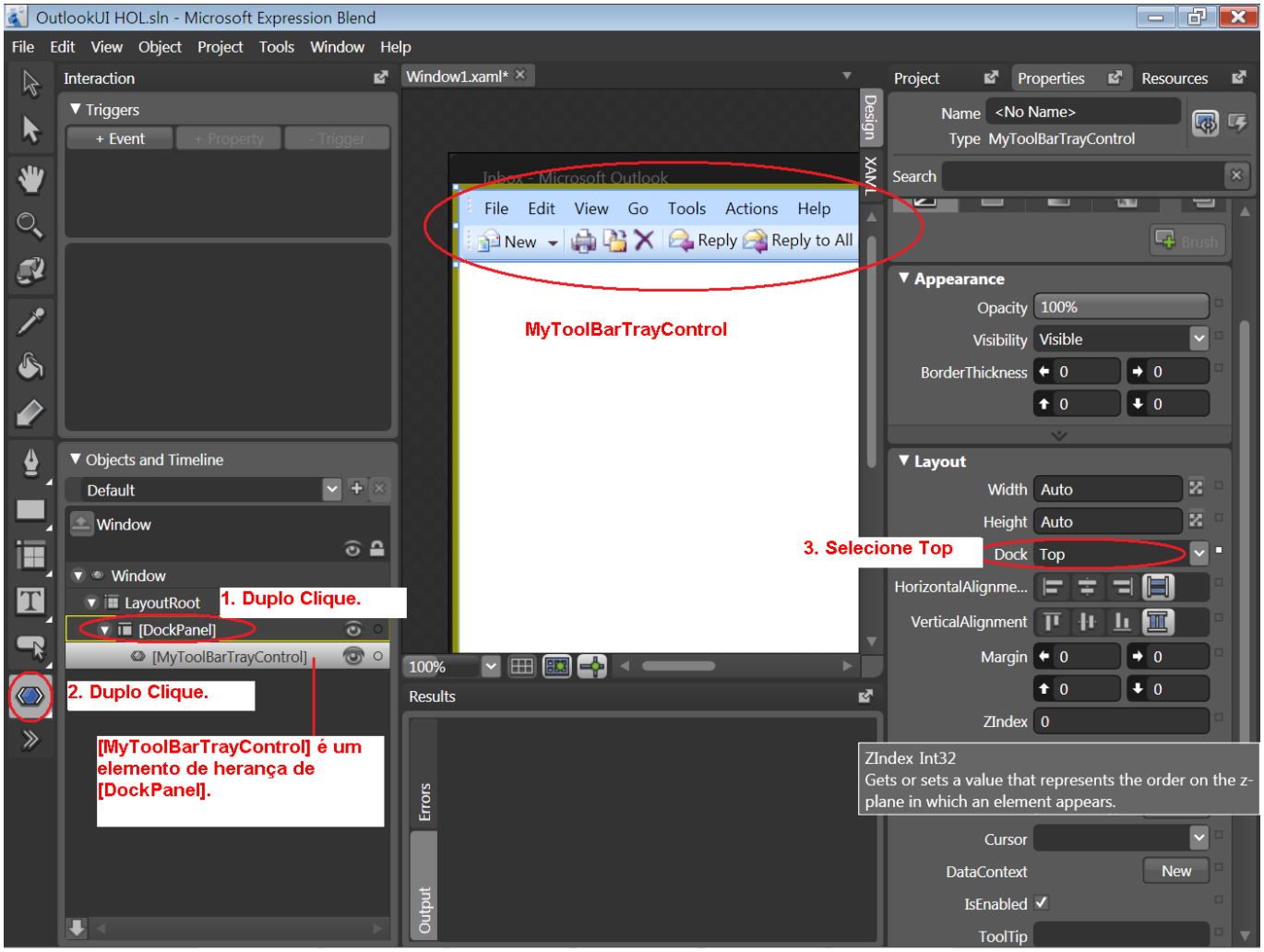


Figura 12 Adicionando um controle MyToolBarTrayControl no [DockPanel].

2. Adicione um **MyStatusBarControl** no **[DockPanel]**.
   Com **[DockPanel]** ativado,
   1) adicione a ferramenta **MyStatusBarControl** no **ToolBox**.
   2) adicione um **MyStatusBarControl** no **[DockPanel]** e defina a propriedade **Dock** como **Bottom**.

3. Adicione um controle **Grid** no **[DockPanel]**. **13**
   Um **Grid** é um recipiente de layout usado com muita freqüência, que consiste em colunas e linhas, e um controle filho pode ser posicionado em uma célula ou abranger mais de uma coluna na respectiva linha. Vamos adicionar um controle **Grid** no **[DockPanel]**:
   1) Com **[DockPanel]** ativado, dê um duplo-clique na ferramenta **Grid** no **ToolBox**.
   2) Restaure a propriedade **Width** do **[Grid]** e defina a propriedade **Dock** como **Bottom**.

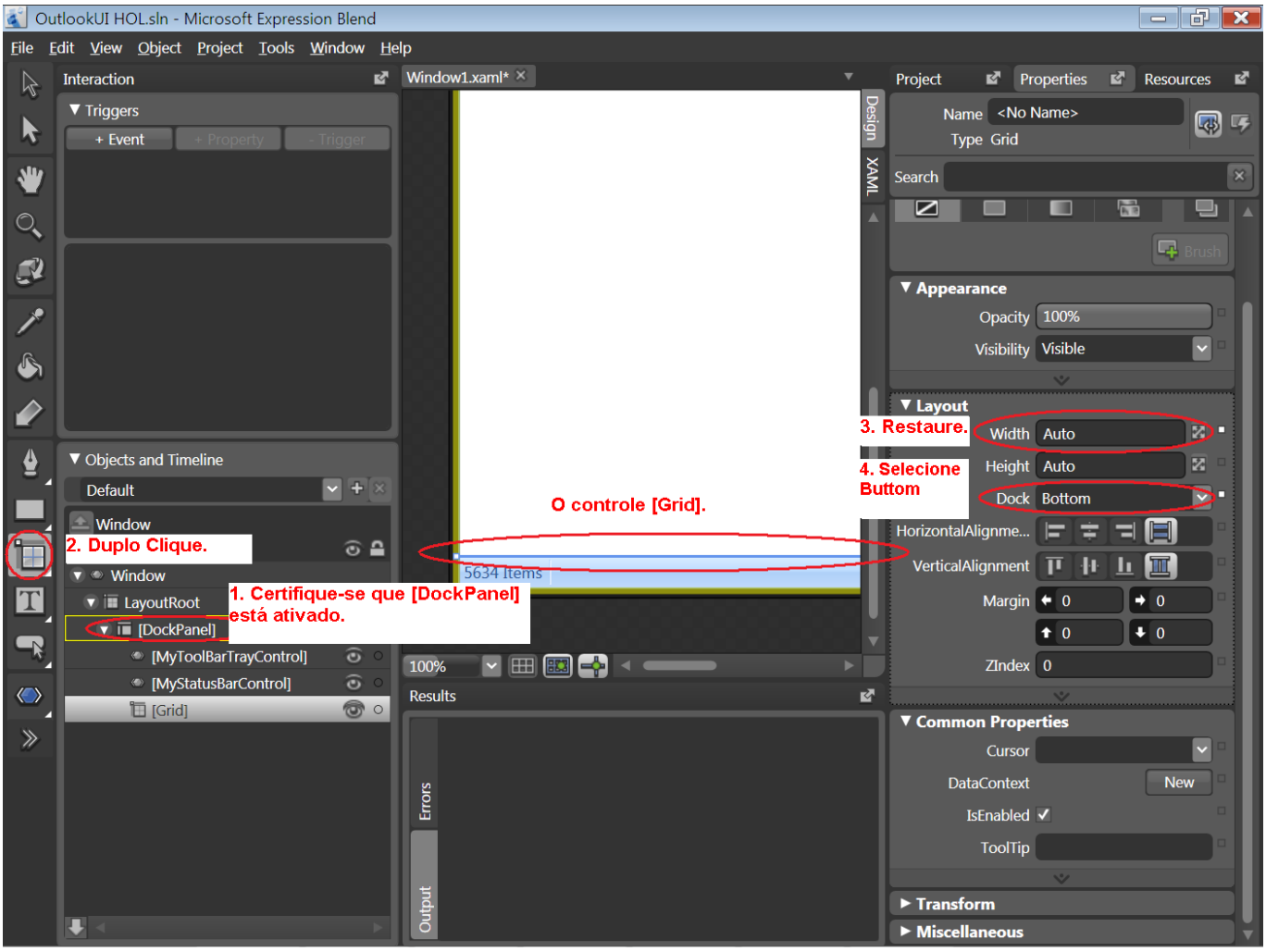


Figura 13 Adicionando um controle Grid no [DockPanel].

4. Defina **[DockPanel].LastChildFill** como **True**. 14
   Neste momento, o controle **[Grid]** não preenche totalmente a área entre o toolbar (barra de ferramentas) e o status bar (barra de status). Para fazer isso, volte para **Properties**, do controle **[DockPanel]**, e marque a propriedade **LastChildFill**.

DockPanel.LastChildFill

Se você definir a propriedade **DockPanel.LastChildFill** para **true**, o último elemento de herança de uma **DockPanel** sempre preencherá o espaço restante, independentemente de qualquer outro valor dock que você definir para o último elemento de herança. No nosso caso, **[Grid]** é adicionado no **[DockPanel]** como o último elemento filho, portanto, ele irá preencher a área entre **[MyToolBarTrayControl]** e **[MyStatusBarControl]**.

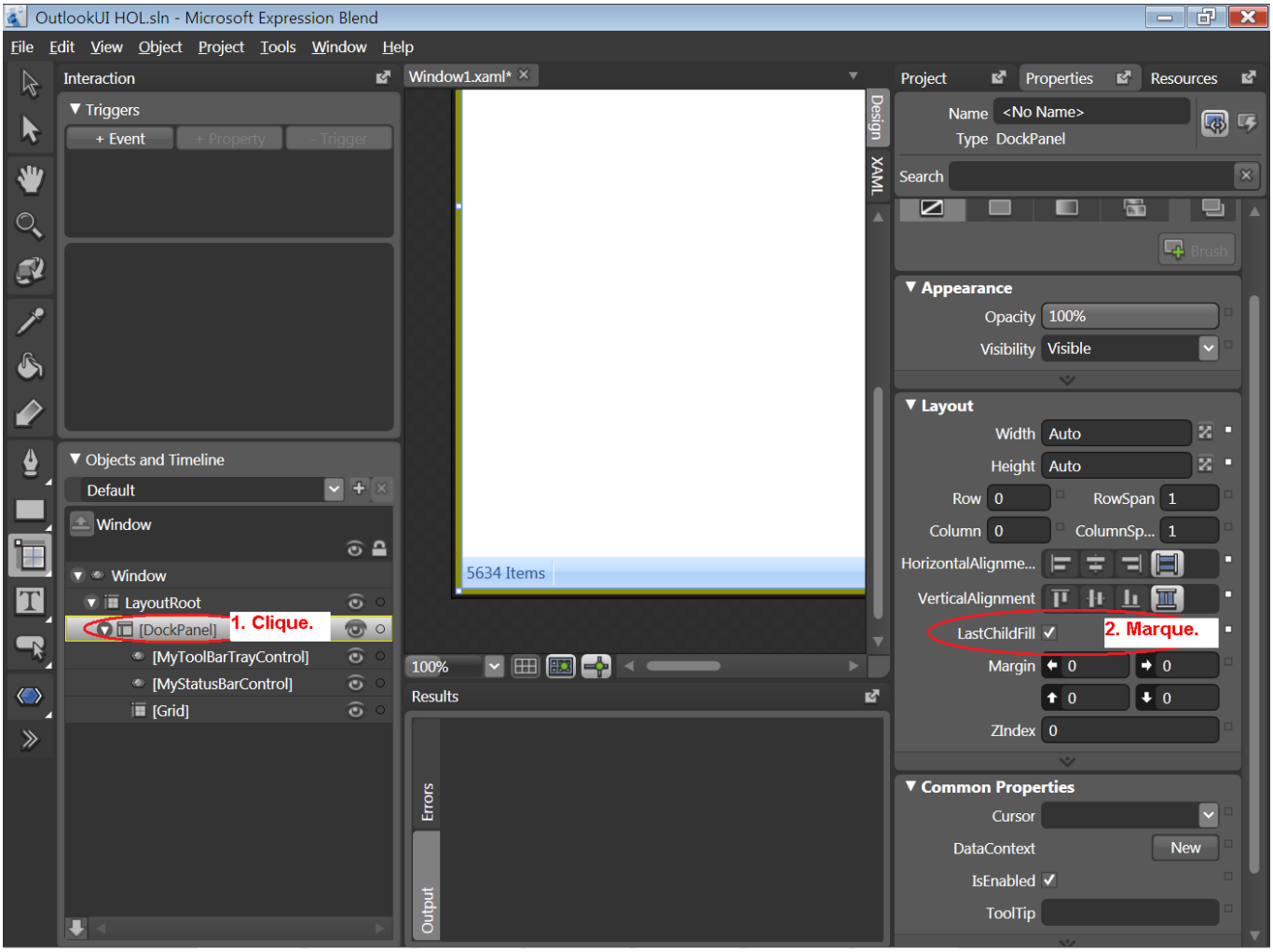


Figura 14 Configurando a propriedade LastChildFill do controle [DockPanel] como True.

5. Agrupe **[Grid]** em um controle **Border** para adicionar uma borda azul ao redor do **[Grid]**. **15**
   Em **Objects and Timeline**, clique com o botão direito do mouse no **[Grid]** para abrir o menu contextual, selecione **Group Into > Border**.

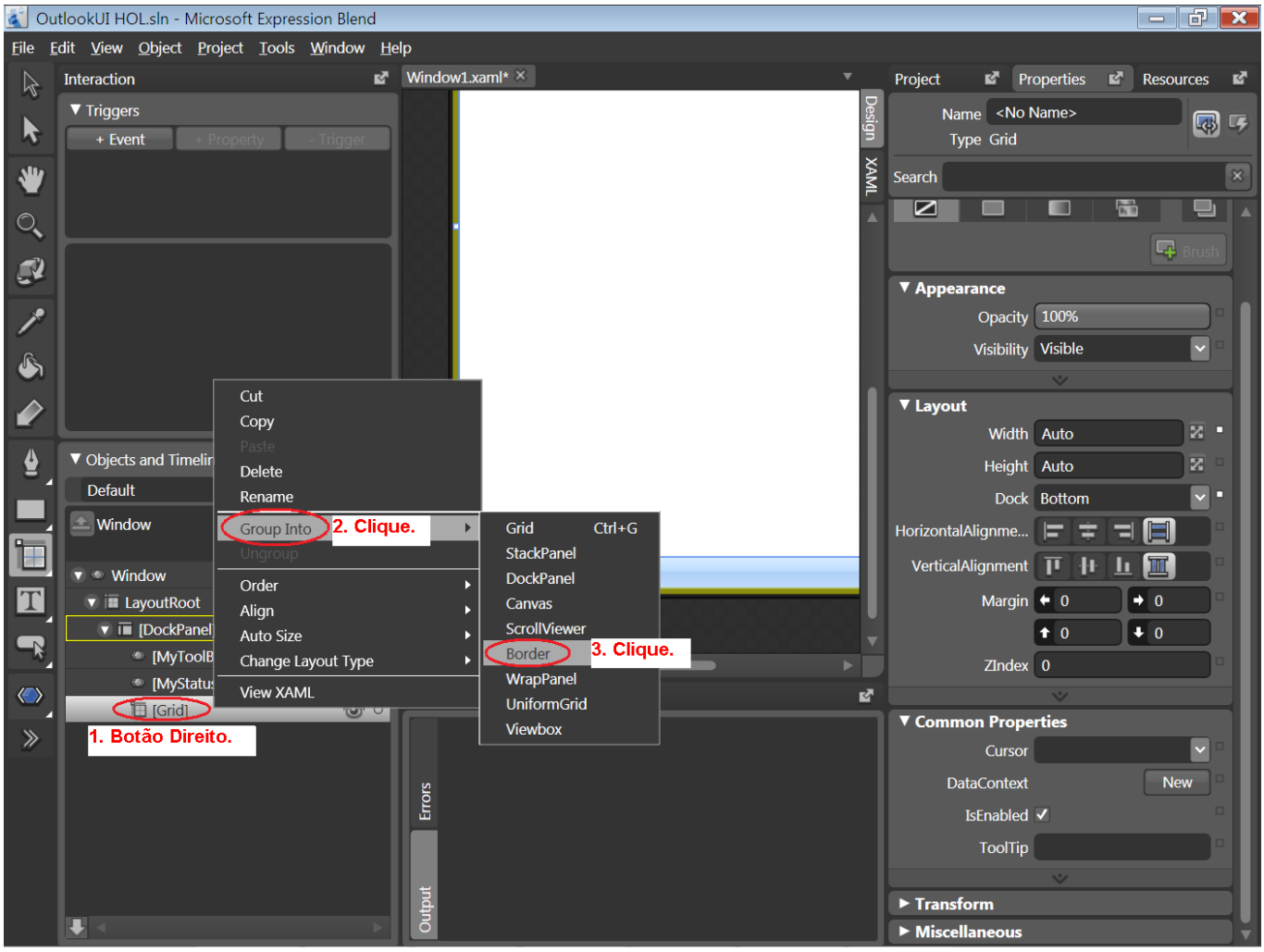


Figura 15 Agrupando o controle [Grid] em um Border.

6. Com **[Border]** selecionado, configure as propriedades do **[Border]** da seguinte maneira: **16**

Brushes

- **BorderBrush = MyBorderBlueGradientBrush**

Appearance

- **BorderThickness: left = 5, right = 5, top = 3, bottom = 5**

Layout

- **Width = Auto, Height = Auto**
- **Dock = Bottom**

A hierarquia dos controles foi atualizada como a seguinte: **[DockPanel]** > **[Border]** > **[Grid]**. É por isso que devemos especificar a propriedade **Dock** para **[Border]** em vez do **[Grid]** agora.

- HorizontalAlignment = Stretch, VerticalAlignment = Stretch
- Margin = 0, 0, 0, 0

Para fins de abrangência, nós alistamos as propriedades de uso geral do controle que está sendo editado, independentemente do valor da propriedade ser o padrão ou não. Você precisa somente prestar atenção às propriedades que são atribuídas aos valores personalizados; estas propriedades estão em negrito.

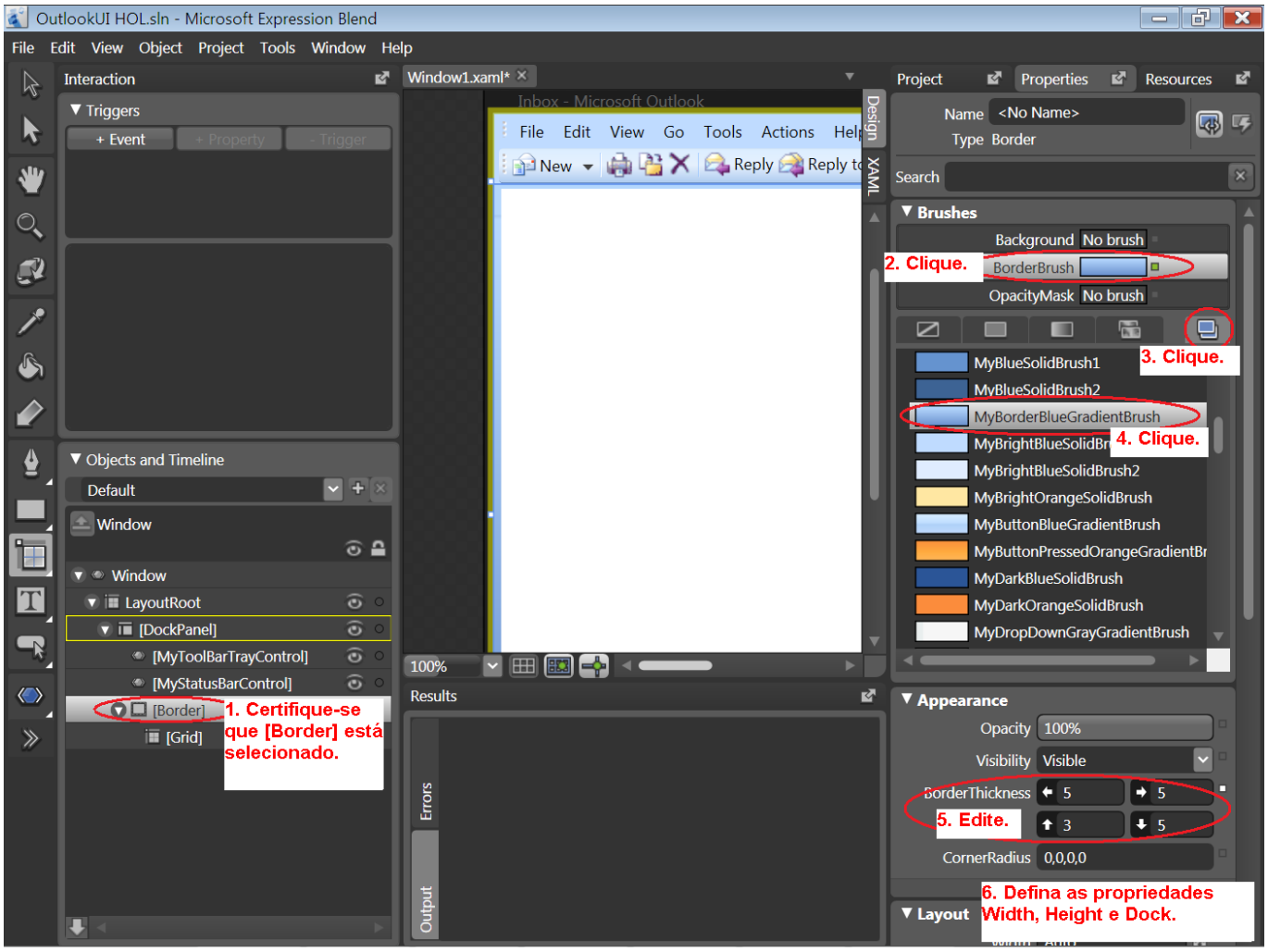


Figura 16 Configurando as propriedades do [Border].

7. Com **[Grid]** selecionado, configure as propriedades do **[Grid]** da seguinte maneira:
   Layout
   - **Width = Auto, Height = Auto**

Tarefa 5: Dividir o [Grid] em Colunas e Linhas & Adicionar o GridSplitter no [Grid]

A próxima etapa é dividir o **[Grid]** em três colunas e duas linhas e adicionar **GridSplitter**s no **[Grid]**.

1. Divida o **[Grid]** em três colunas e duas linhas. 17
    1) Em **Objects and Timeline**, dê um duplo-clique no **[Grid]** para ativá-lo.

Pressione **F4** para obter a imagem inteira do **[Grid]**. Para retornar ao modo de exibição original, pressione **F4** novamente.

    2) Clique na ferramenta **Selection** , no **ToolBox**.
    3) Sobre o artboard, mova o ponteiro do mouse sobre a área da régua azul grossa na parte superior do **[Grid]**. Uma coluna de régua laranja seguirá o ponteiro do mouse e indicará onde uma nova coluna divisória será colocada, caso você clique. Clique para criar um novo divisor de coluna. Um divisor de coluna azul aparece dentro do **[Grid]**.
    4) Crie um segundo divisor de coluna.
    5) Mova o ponteiro do mouse sobre a área da régua azul grossa à esquerda do **[Grid]**, clique para criar uma linha divisória.

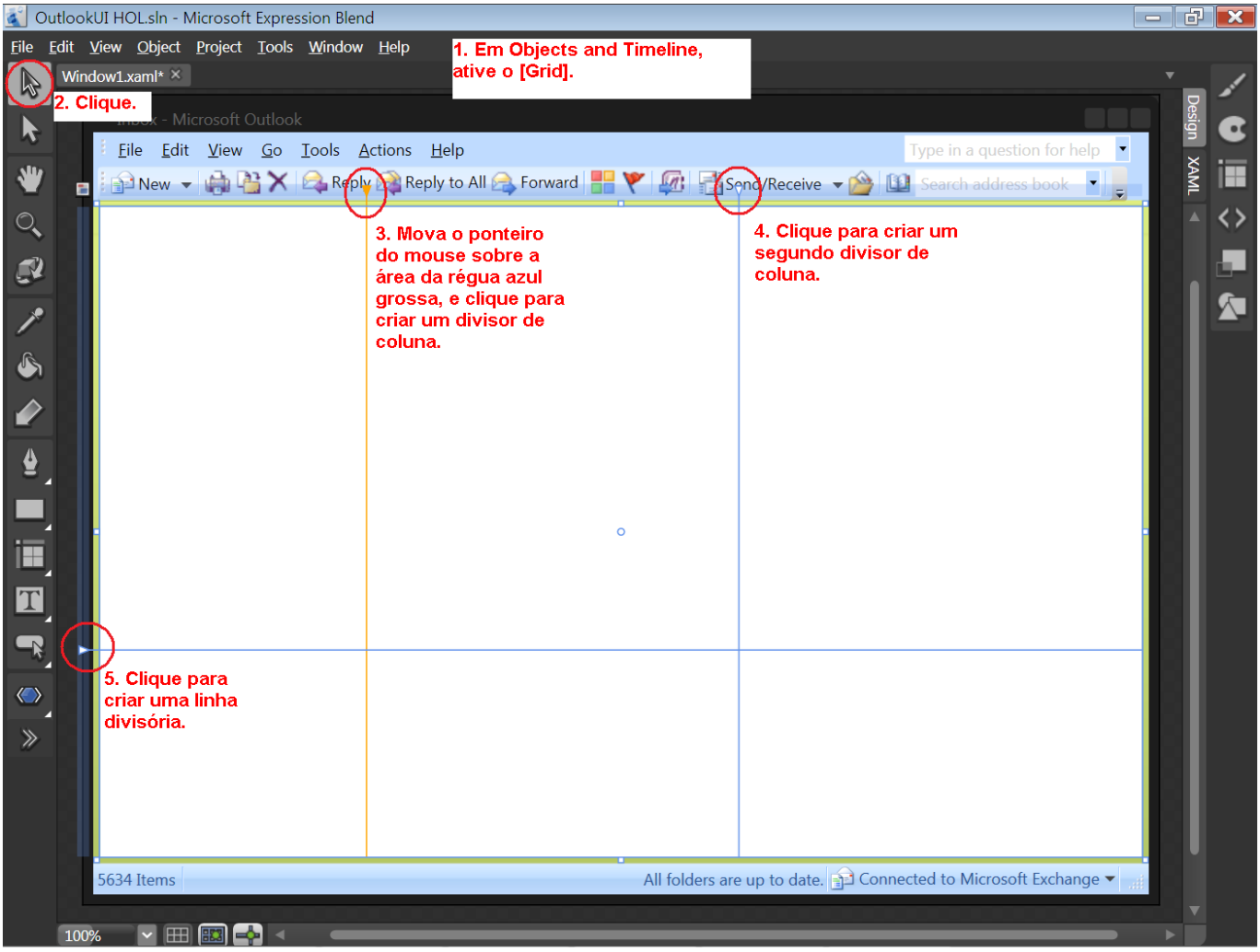

OutlookUI HOL.sln - Microsoft Expression Blend
File Edit View Object Project Tools Window Help
1. Em Objects and Timeline, ative o [Grid].
Window1.xaml*
2. Clique.
Microsoft Outlook
File Edit View Go Tools Actions Help
Type in a question for help
New Reply Reply to All Forward Send/Receive Search address book
3. Mova o ponteiro do mouse sobre a área da régua azul grossa, e clique para criar um divisor de coluna.
4. Clique para criar um segundo divisor de coluna.
5. Clique para criar uma linha divisória.
5634 Items
All folders are up to date.
Connected to Microsoft Exchange
100%
Design
XAML

Figura 17 Criando dois divisores de coluna e uma linha divisória para o [Grid].

Caso queiramos que o usuário possa ajustar a largura da coluna e a altura da linha em tempo de execução, devemos adicionar **GridSplitter**s no **[Grid]**.

2. Adicione um controle **GridSplitter** vertical no lado esquerdo da segunda coluna do **[Grid]**. **18**
    1) Ative o **[Grid]** se ele não estiver ativo.
    2) Adicione a ferramenta **GridSplitter**, do **Asset Library**, no **ToolBox**.
    Agora, quando você abre a caixa de diálogo de **Asset Library**, a guia atual será **Custom Controls**. Caso a ferramenta **GridSplitter** seja localizada na categoria **Controls**, primeiro você deve alternar para a guia Controls a fim de usar a ferramenta **GridSplitter**.
    3) Clique na ferramenta **GridSplitter**, no **ToolBox**.
    4) Sobre o artboard, desenhe um controle **GridSplitter** vertical no lado esquerdo da segunda coluna.
    5) Configure as propriedades do **[GridSplitter]** da seguinte maneira:
        - **Name = myMainGridFirstColumnSplitter**

Nome do controle exclusivo para referência programática

Em WPF, nem todos os controles têm que ser nomeado. Em **Objects and Timeline**, controles sem nome são mostrados como **[Control]**.
No entanto, se você deseja se referir a um controle programado em código VB/C#, você deve dar a esse controle um nome exclusivo. No nosso caso, como você verá, nós devemos escrever um código de tratamento de evento para os eventos dos **GridSplitter**s para ativar um comportamento mais avançado, que é por isso que ele deve ser nomeado.

Os códigos C# e XAML diferenciam maiúsculas de minúsculas

Como ambos os códigos XAML e C# diferenciam maiúsculas de minúsculas, você deve prestar atenção em letras maiúsculas e minúsculas enquanto edita arquivos XAML e seus respectivos códigos C#.

Brushes

- **Background = MyBorderBlueGradientBrush**

Layout

- **Width = 5, Height = Auto**
- **Row = 0** (índice baseado em zero), **RowSpan = 2, Column = 1, ColumnSpan = 1**

Grid.RowSpan e Grid.ColumnSpan

A propriedade **Grid.RowSpan** obtém ou define um valor que indica o número total de linhas que abrangem o conteúdo de herança dentro de um **Grid**. A propriedade **Grid.ColumnSpan** obtém ou define um valor que indica o número total de colunas que abrangem o conteúdo de herança dentro de um **Grid**.

- **HorizontalAlignment = Left** (isso configura o elemento **[GridSplitter]** para alinhar a borda esquerda da segunda coluna), **VerticalAlignment = Stretch**
- **Margin = 0, 0, 0, 0**

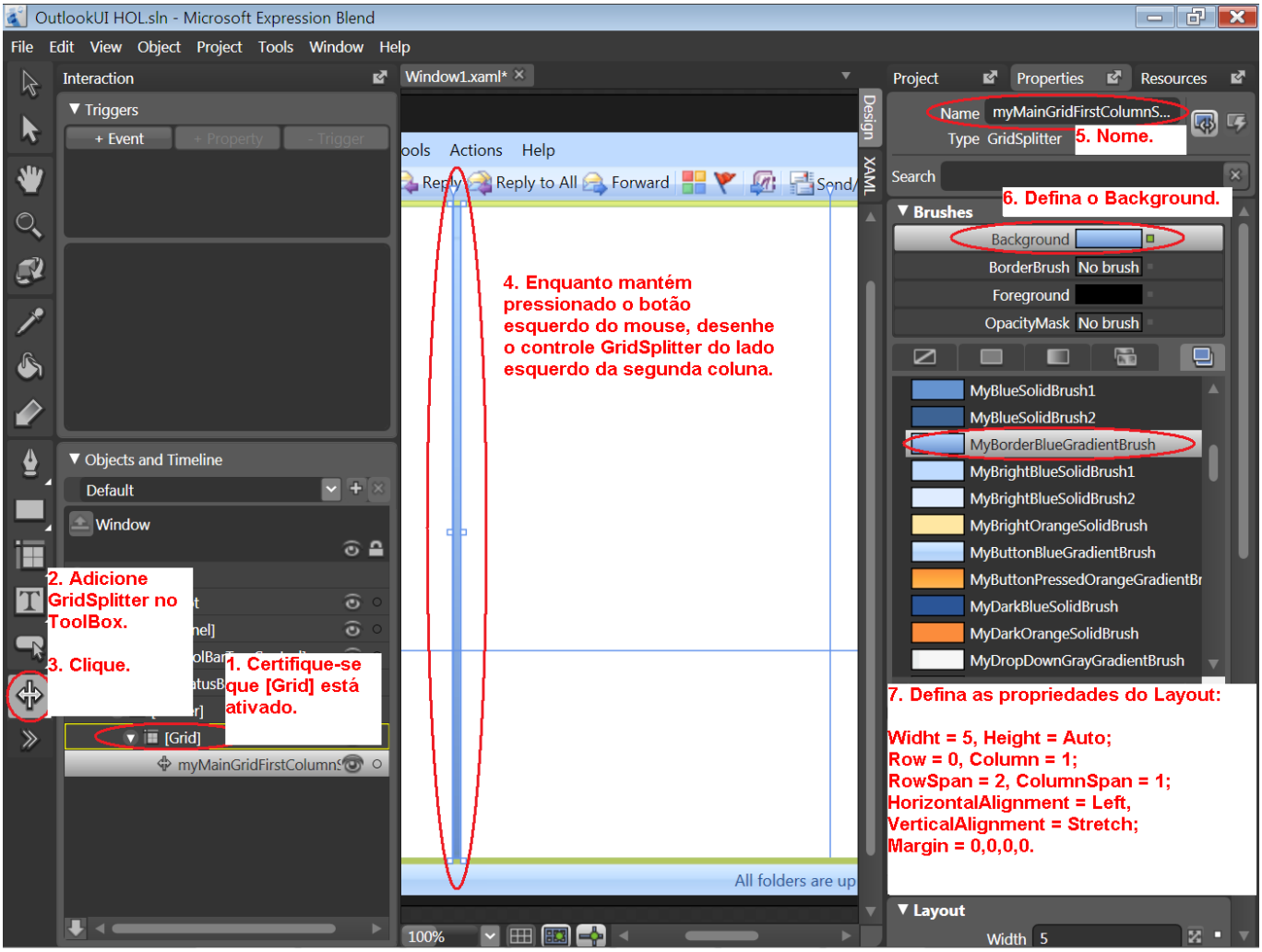


Figura 18 Adicionando um controle GridSplitter no [Grid] do lado esquerdo da segunda coluna.

3. Adicione um segundo controle vertical de **GridSplitter** do lado direito da segunda coluna do **[Grid]**, e adicione então um controle horizontal de **GridSplitter** na primeira coluna e na parte inferior da primeira linha do **[Grid]**.
   Como você já sabe adicionar um **GridSplitter** no **[Grid]**, você não precisa perder tempo no segundo e no terceiro item. Nós preparamos trechos XAML que você pode copiar e então colar diretamente no arquivo **Window1.xaml**. As seguintes etapas mostram como fazer isso:

   1) Alterne para o **modo XAML** do **[Grid]**. **19**
      Em **Objects and Timeline**, clique com o botão direito do mouse no **[Grid]** para abrir o menu contextual, escolha **View XAML** para alternar para o **modo XAML**.

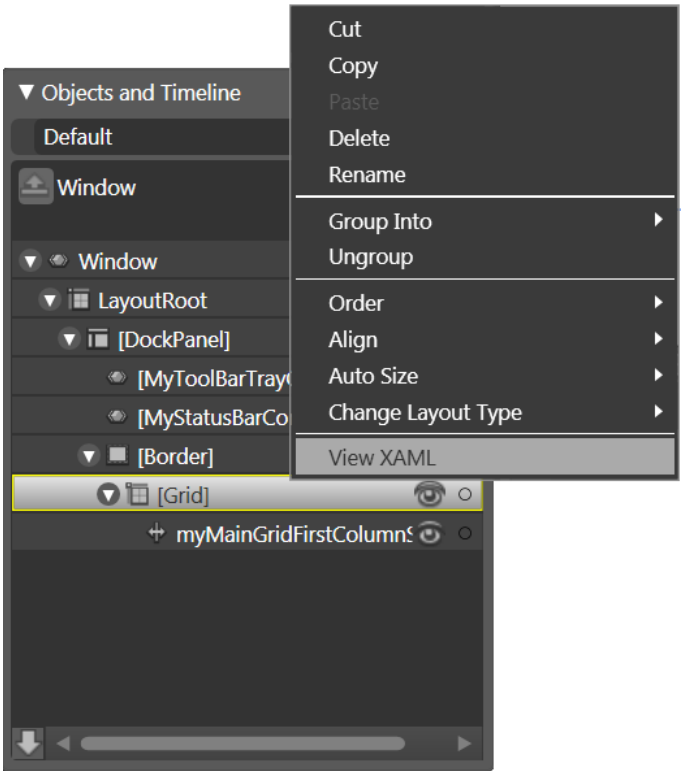


Figura 19 Alternando para o modo XAML do [Grid].

Como pode ver, o código XAML do elemento **[Grid]** é realçado em azul no seu plano de fundo. Pressione **F4** para obter uma exibição maior do arquivo.

2) Adicione dois elementos **GridSplitter** no elemento **[Grid]**. **20**

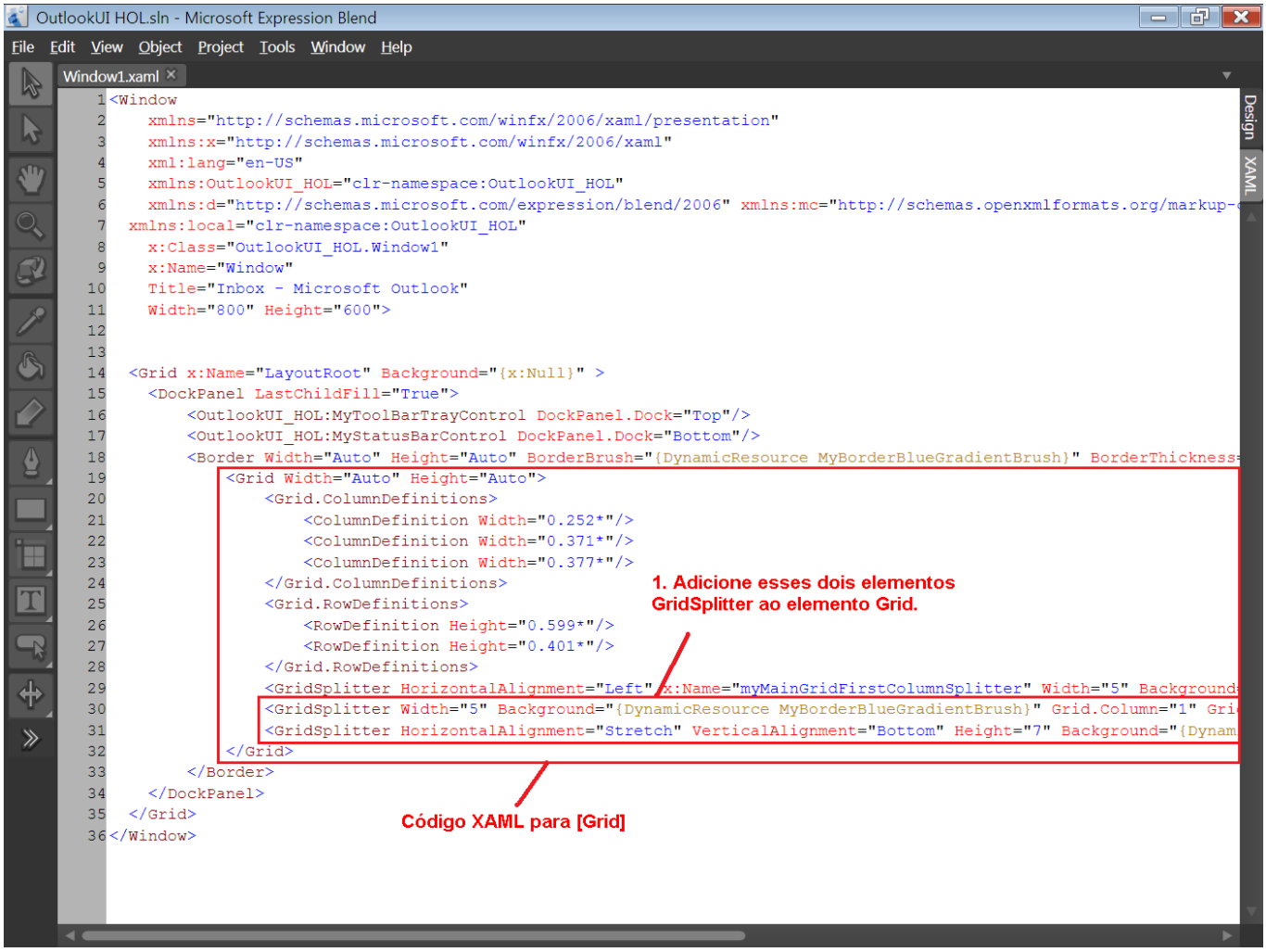

OutlookUI HOL.sln - Microsoft Expression Blend
File Edit View Object Project Tools Window Help
Window1.xaml
1. Adicione esses dois elementos GridSplitter ao elemento Grid.
Código XAML para [Grid]

Figura 20 Adicionando dois elementos GridSplitter no [Grid].

Gerando XAML no modo Design ou no modo XAML

No modo **Design**, você pode criar seu aplicativo visualmente e deixar o Expression Blend gerar o XAML para você. Editar o XAML diretamente pode resultar em erros de análise gramatical em seu aplicativo que você precisará corrigir antes que o Expression Blend possa corretamente exibir seus documentos sobre o artboard no modo **Design**, ou antes de criar e executar o aplicativo. Com esse cuidado em mente, trabalhar entre os modos de exibição **Design** e **XAML** pode ser um método eficaz para aprender os fundamentos do XAML.

No modo **XAML**, você pode digitar novos elementos no código, ou você pode selecionar o código existente e então recortar ou copiar a partir dele ou colá-lo exatamente como faria em um programa de processamento de palavras. Você também pode ir para linhas específicas no XAML ou localizar e substituir texto, usando os comandos **Go To**, **Find**, **Find Next** e **Replace** do menu **Edit**.

As alterações feitas no modo **XAML** são atualizadas automaticamente no modo **Design**, e as alterações feitas no modo **Design** são atualizadas automaticamente no modo **XAML**.

O Expression Blend atualiza os erros de sintaxe no painel **Results** quando você salva o arquivo ou quando você alterna entre os modos de exibição (modo **Design** e modo **XAML**).

Adicione o seguinte código XAML para esses dois elementos **GridSplitter** após o primeiro elemento **GridSplitter**:

basic5.txt

```
<GridSplitter Width="5" Background="{DynamicResource MyBorderBlueGradientBrush}" Grid.Column="1" Grid.RowSpan="2"/>
```

<GridSplitter HorizontalAlignment="Stretch" VerticalAlignment="Bottom" Height="7" Background="{DynamicResource MyHGridSplitterBlueGradientBrush}"/>

Alterne para o **modo Design** e pressione **F4** novamente para retornar à interface com o usuário normal do **Expression Blend**.

É hora para criar e executar o aplicativo! Pressione **F5** e espere até que seu primeiro aplicativo WPF apareça. Seu aplicativo deve ter a aparência da figura 21. Mova o mouse sobre a barra de ferramentas, clique nos menus, nos botões, nas imagens e nas caixas de texto, arraste o **GridSplitter** vertical para a esquerda e para a direita, e arraste o **GridSplitter** horizontal para cima e para baixo.

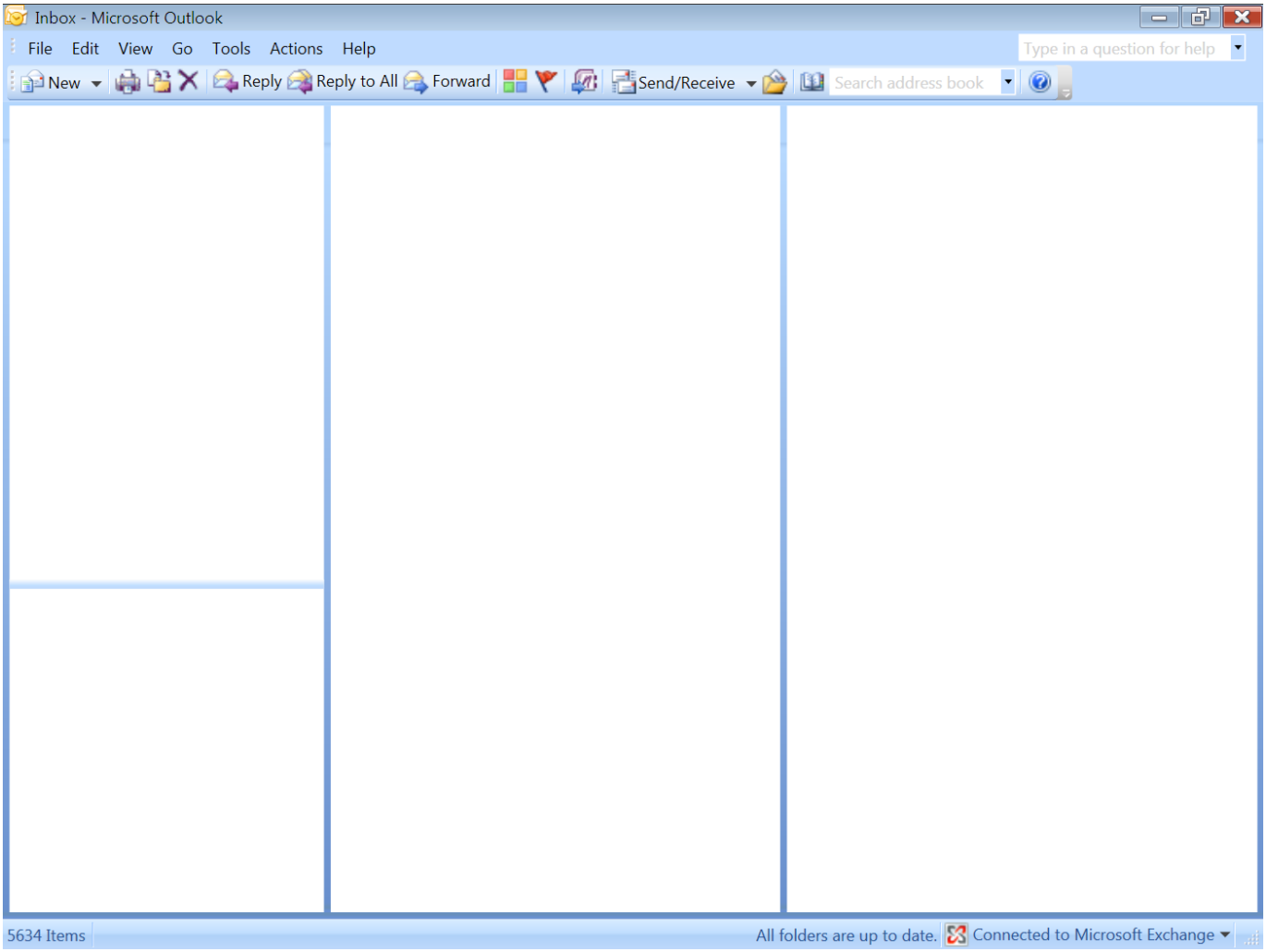


Figura 21 A interface com o usuário do aplicativo após a tarefa 5.

Tarefa 6: Adicionar um StackPanel e seus controles de herança à célula inferior esquerda do [Grid]

Nesta tarefa, você aprenderá a adicionar um controle **StackPanel** e seus controles de herança à célula inferior esquerda do **[Grid]** (veja figura 22).

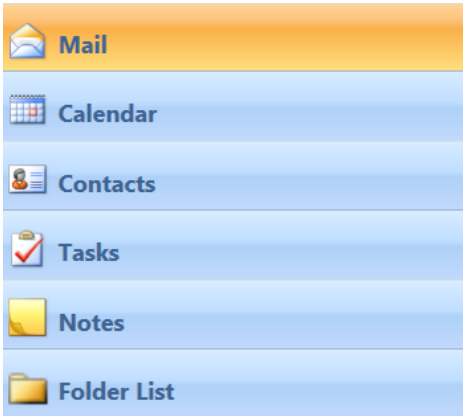


Figura 22 O StackPanel com seus controles de herança.

A estrutura do elemento acima é semelhante ao seguinte (veja figura 23): Um controle **StackPanel** vertical situa-se no nível superior; ele contém 6 botões. Cada botão contém novamente um controle **StackPanel** horizontal que é usado para alinhar os controles **Image** e **TextBlock**.

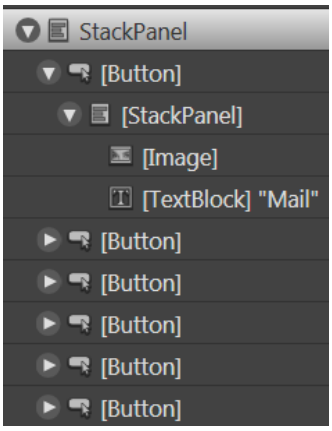


Figura 23 Estrutura de controle hierárquica do StackPanel e de seus controles de herança.

1. Adicione um controle **StackPanel** no **[Grid]** e configure suas propriedades da seguinte maneira: 24
   Layout
   - **Width = Auto, Height = Auto**
   - **Row = 1,** RowSpan = 1, Column = 0, ColumnSpan = 1
   - Orientation = Vertical
   - **HorizontalAlignment = Stretch, VerticalAlignment = Top**
   - **Margin = 0, 0, 0, 0**

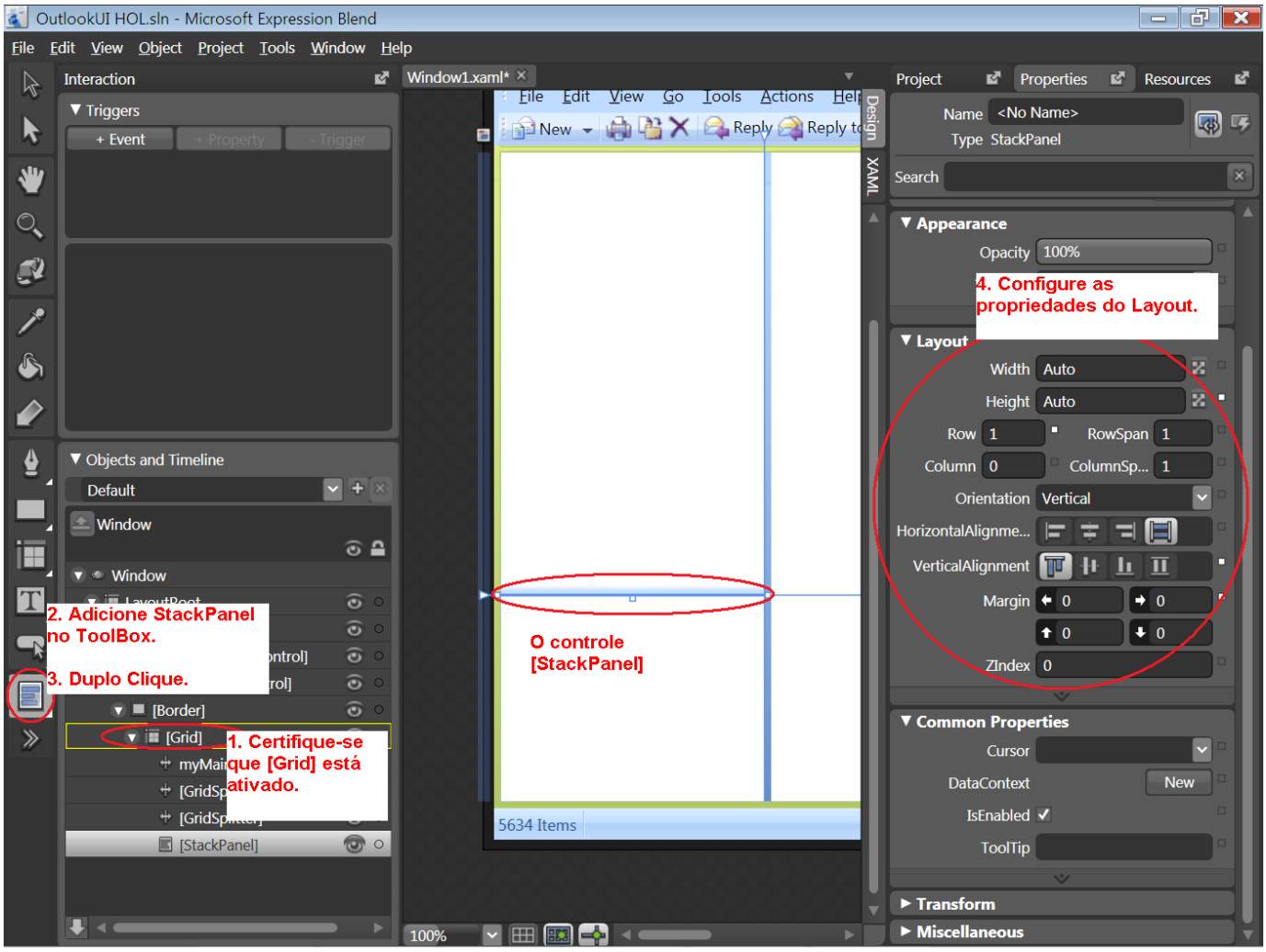


Figura 24 Adicionando um controle StackPanel no controle [Grid].

2. Adicione um controle **Button** no **[StackPanel]** e configure suas propriedades da seguinte maneira: **25**

Brushes

- **Background = MySelectedButtonOrangeGradientBrush**
- **BorderBrush = MyBlueSolidBrush2**

Appearance

- **BorderThickness: left = 0, right = 0, top = 0.2, bottom = 0.2**

Layout

- Width = Auto, **Height = 35**
- HorizontalAlignment = Stretch, VerticalAlignment = Stretch
- Margin = 0, 0, 0, 0
- **HorizontalContentAlignment = Left,** VerticalContentAlignment = Center

  Para ver essas duas propriedades você precisar expandir a categoria **Layout**, clicando na seta sob a propriedade **ZIndex**.

Propriedades Control.HorizontalContentAlignment e Control.VerticalContentAlignment

A propriedade **Control.HorizontalContentAlignment** obtém ou define o alinhamento horizontal do conteúdo de um controle, semelhante a **FrameworkElement.HorizontalAlignment**, ele tem igualmente quatro valores possíveis: **Left**, **Center**, **Right** e **Stretch**.

A propriedade **Control.VerticalContentAlignment** obtém ou define o alinhamento horizontal do conteúdo de um controle, semelhante a **FrameworkElement.VerticalAlignment**, ele tem igualmente quatro valores possíveis: **Top**, **Center**, **Bottom** e **Stretch**.

- Padding: **left = 3,** right = 1, top = 1, bottom = 1
  Para ver esta propriedade você também precisa expandir a categoria **Layout**.

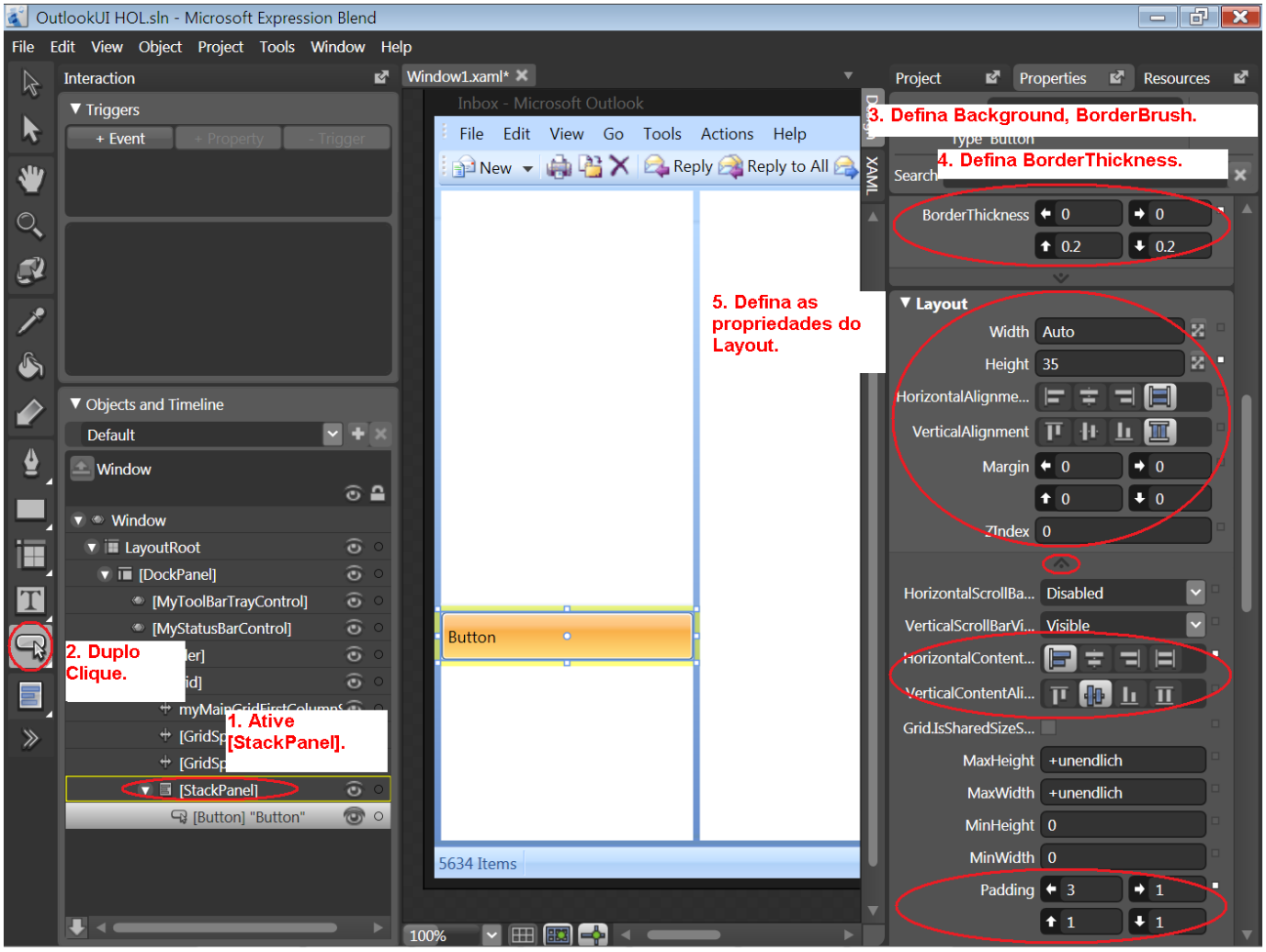


Figura 25 Adicionando um controle Button no [StackPanel].

Para que os controles **[Button]** tenham a mesma aparência do MS Outlook 2007, nós definimos nosso próprio estilo, chamado **MyBottomLeftButtonStyle**, para estes botões.

Style (estilo)

Um **Style** permite que você defina valores de propriedades diferentes para diferentes estados em que um controle se encontra. Por exemplo, no caso de um botão, nós podemos definir sua propriedade **Background** para azul em seu estado **Default**, e nós também podemos definir sua propriedade **Background** para laranja caso o usuário mova o mouse sobre ele, e finalmente, a propriedade **Background** pode ser vermelha, quando o usuário clica nele.

Faça o seguinte para definir a propriedade **Style** do **[Button]** para **MyBottomLeftButtonStyle**:
Miscellaneous

- **Style = MyBottomLeftButtonStyle** **26**
  Expanda a categoria **Miscellaneous**, clique no quadrado pequeno ao lado da propriedade **Style** para abrir o menu contextual, clique em **Local Resource** e selecione **MyBottomLeftButtonStyle**.

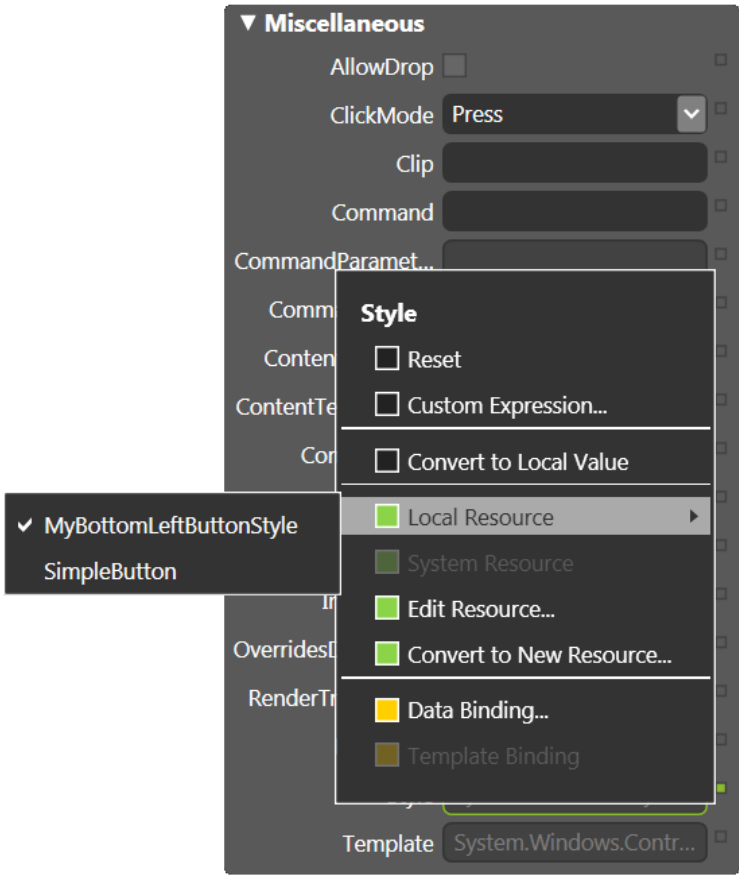


Figura 26 Ajustando a propriedade Style do controle [Button] para MyBottomLeftButtonStyle.

3. Adicione um controle **StackPanel** no **[Button]**.

   Primeiramente ative **[Button]**, depois dê um duplo-clique na ferramenta **StackPanel** no **ToolBox**, e em seguida, configure as propriedades do **[StackPanel]** da seguinte maneira:
   Layout

   - **Width = Auto, Height = 26**
   - **Orientation = Horizontal**

   O controle **[StackPanel]** adicionado nessa etapa deve conter dois elementos de herança: um controle **Image** e um controle **TextBlock**.
4. Adicione um controle **Image** no **[StackPanel]** criado na etapa 3.
   1) Ative **[StackPanel]** localizado no **[Button]**.
   2) Alterne para a guia **Project**.
   3) Em **OutlookUI HOL**, expanda a pasta **graphics** e dê um duplo-clique no arquivo de imagem **mail.ico**.
   4) Configure as propriedades do controle **[Image]** da seguinte maneira:
      Layout

      - **Width = 21, Height = 21**
      - **Margin**: left = 0, right = 0, **top = 5,** bottom = 0

5. Adicione um controle **TextBlock** no **[StackPanel]** criado na etapa 3.
   1) Ative **[StackPanel]**, localizado no **[Button]** se ele não estiver ativo.
   2) Selecione a ferramenta **TextBlock** no **ToolBox**: 27
      Clique com o botão direito do mouse no pequeno triângulo incluído na ferramenta **TextBox**, no **ToolBox**, para obter uma lista dos outros recursos semelhantes, e selecione, então, **TextBlock**.

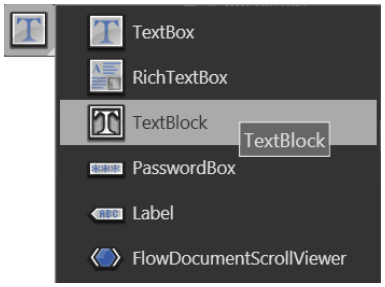


Figura 27 Selecionando a ferramenta TextBlock.

   3) Dê um duplo-clique na ferramenta **TextBlock**, no **ToolBox**, para adicionar um controle **TextBlock** no **[StackPanel]**.
   4) Configure as propriedades do controle **[TextBlock]** da seguinte maneira:

O **Expression Blend** não indica corretamente o **Properties editor** depois que você adiciona um controle **TextBlock**. Apenas selecione outro controle, e selecione, então, o controle **TextBlock** outra vez para editar suas propriedades. Lembre-se desse truque.

Brushes

- **Foreground = MyBlueSolidBrush2**

Layout

- **HorizontalAlignment = Left, VerticalAlignment = Center**
- **Margin: left = 4,** right = 0, **top = 7,** bottom = 0

Common Properties

- **Text = Mail**

Text

- **FontWeight = Bold**

6. Adicione outros cinco botões.
   Como os seis botões têm a mesma estrutura e só diferem nas propriedades **Image.Source** e **TextBlock.Text**, podemos usar a técnica de Copiar e Colar para economizar muito trabalho:
   1) Em **Objects and Timeline**, clique com o botão direito do mouse no **[Button]** para obter o menu contextual indicado, e clique em **Copy**.
   2) Ative **[StackPanel]**, localizado no **[Button]**.
   3) Clique com o botão direito do mouse no **[StackPanel]** para obter o menu contextual indicado, e clique em **Paste**.
   4) Repita a etapa 3) para as outras quatro vezes.
   5) Defina a propriedade **Background** como **MyButtonBlueGradientBrush** e restaure a propriedade **Width** dos últimos cinco botões. 28

Uma vantagem no **Expression Blend** é que você pode configurar propriedades comuns de vários controles selecionados de uma vez.

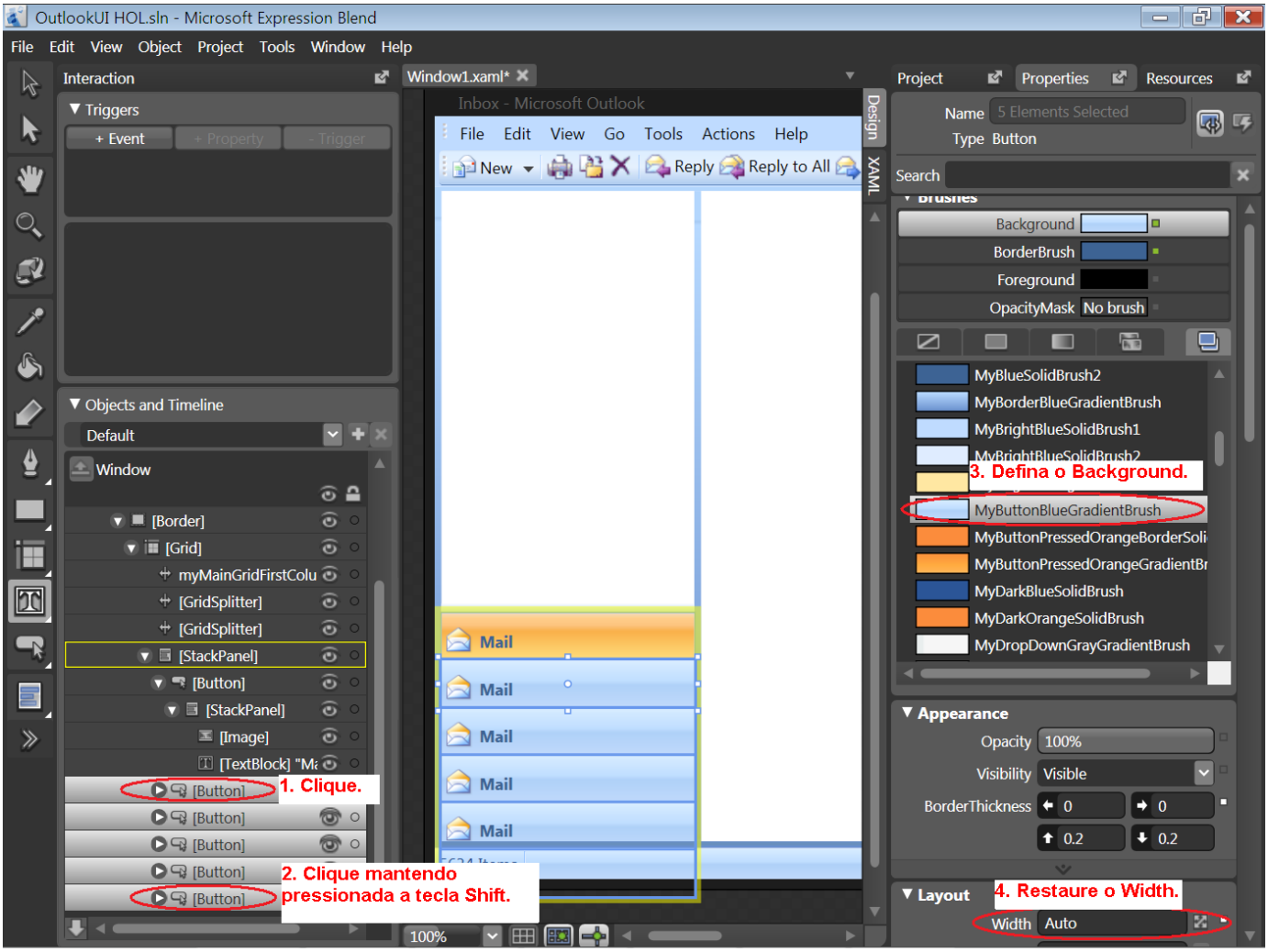


Figura 28 Definindo as propriedades Background e Width dos últimos cinco botões.

Sugerimos que você faça a seguinte etapa opcional após ter concluído o curso básico ou o curso avançado: Corrigir as propriedades **Image.Source** e **TextBlock.Text** dos últimos cinco botões. Os valores das propriedades são:

- Image.Source = graphics\calendar.ico, TextBlock.Text = Calendar
- Image.Source = graphics\contacts.ico, TextBlock.Text = Contacts
- Image.Source = graphics\tasks.ico, TextBlock.Text = Tasks
- Image.Source = graphics\notes.ico, TextBlock.Text = Notes
- Image.Source = graphics\folder.ico, TextBlock.Text = Folder List

Crie e execute o projeto (**F5**). Arraste o **GridSplitter** horizontal para cima e para baixo; como pode ver, há três problemas com o comportamento do controle **[StackPanel]** dinâmico:

1. Com grande probabilidade de apenas parte do botão, da parte inferior, ser exibido.
2. O **GridSplitter** horizontal atualiza sua posição vertical de pixel por pixel quando você arrasta-o, de modo que os botões nem sempre são exibidos totalmente.
3. O controle **[StackPanel]** não usa a parte inferior quando você arrasta o **GridSplitter** horizontal alto o suficiente.

O primeiro problema pode ser solucionado tornando a segunda linha do **[Grid]** alto o suficiente para permitir que todo o **[StackPanel]** seja exibido quando o aplicativo for inicializado (consulte as etapas 1-4 descritas abaixo).

Para resolver o segundo problema, faça o seguinte: Vá para o **Properties editor** do controle **GridSplitter** horizontal, da categoria **Common Properties**; defina a propriedade **DragIncrement** para o valor **Height** dos nossos botões, ou seja, **35**.

Data binding (ligação de dados) mostra sua força quando tentamos resolver o terceiro problema.

Data binding (ligação de dados)

Data binding é o processo de conectar um data source (fonte de dados) a componentes de interface com o usuário. Isso significa que sempre que os dados forem alterados, os componentes da interface serão opcionalmente refletidos para essas alterações e vice-versa. Os dois componentes essenciais de um data binding são uma fonte e um destino. A fonte pode ser de um data source XML, outro controle, etc., e o destino é um controle.

No nosso caso, queremos vincular a propriedade **Grid.Row[1].MaxHeight** ao valor da propriedade **StackPanel.ActualHeight**:

1. Selecione o controle **[Grid]**.
2. No **Properties editor**, expanda a categoria **Layout**. Localize a propriedade **RowDefinitions (Collection)**, clique no botão ; uma caixa de diálogo, em seguida, é exibida.
3. Defina a propriedade **Height** da primeira linha como **0.5***. **29**

Star-sizing

Star-sizing indica que os tamanhos das linhas são proporcionais entre eles. Por exemplo, um **Height** de "3*" produzirá uma linha que seja três vezes o tamanho de uma linha cujo **Height** seja definido como "*".

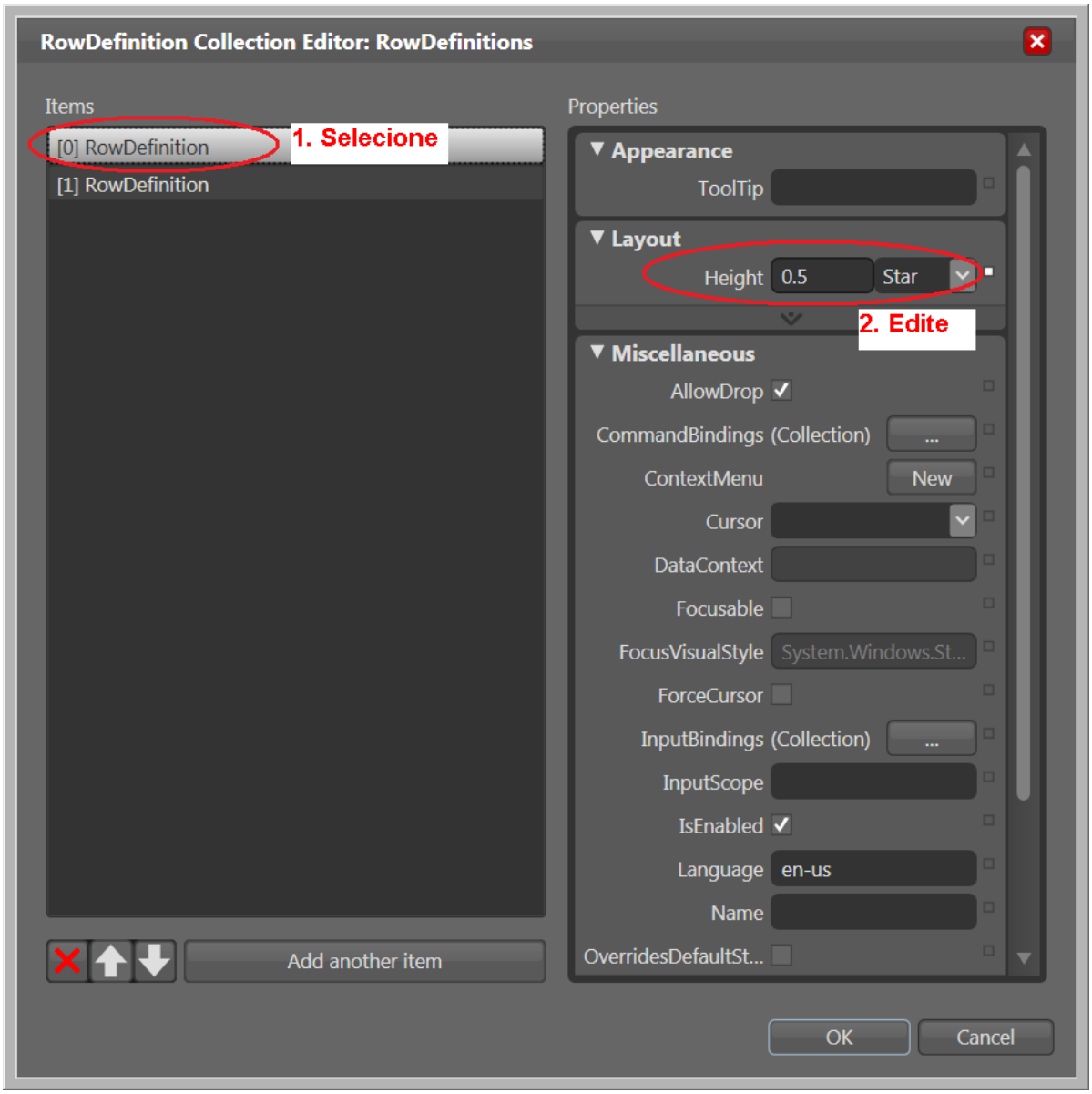


Figura 29 Configurando a propriedade Height da primeira linha como 0.5*.

4. Configure as propriedades da segunda linha **[1] RowDefinition** da seguinte maneira: **30**
   - **Height = 0.5***
   - **MinHeight = 35,** este é o valor de **Height** do **Button**, a fim de evitar que o **StackPanel** comece escondido na tela.

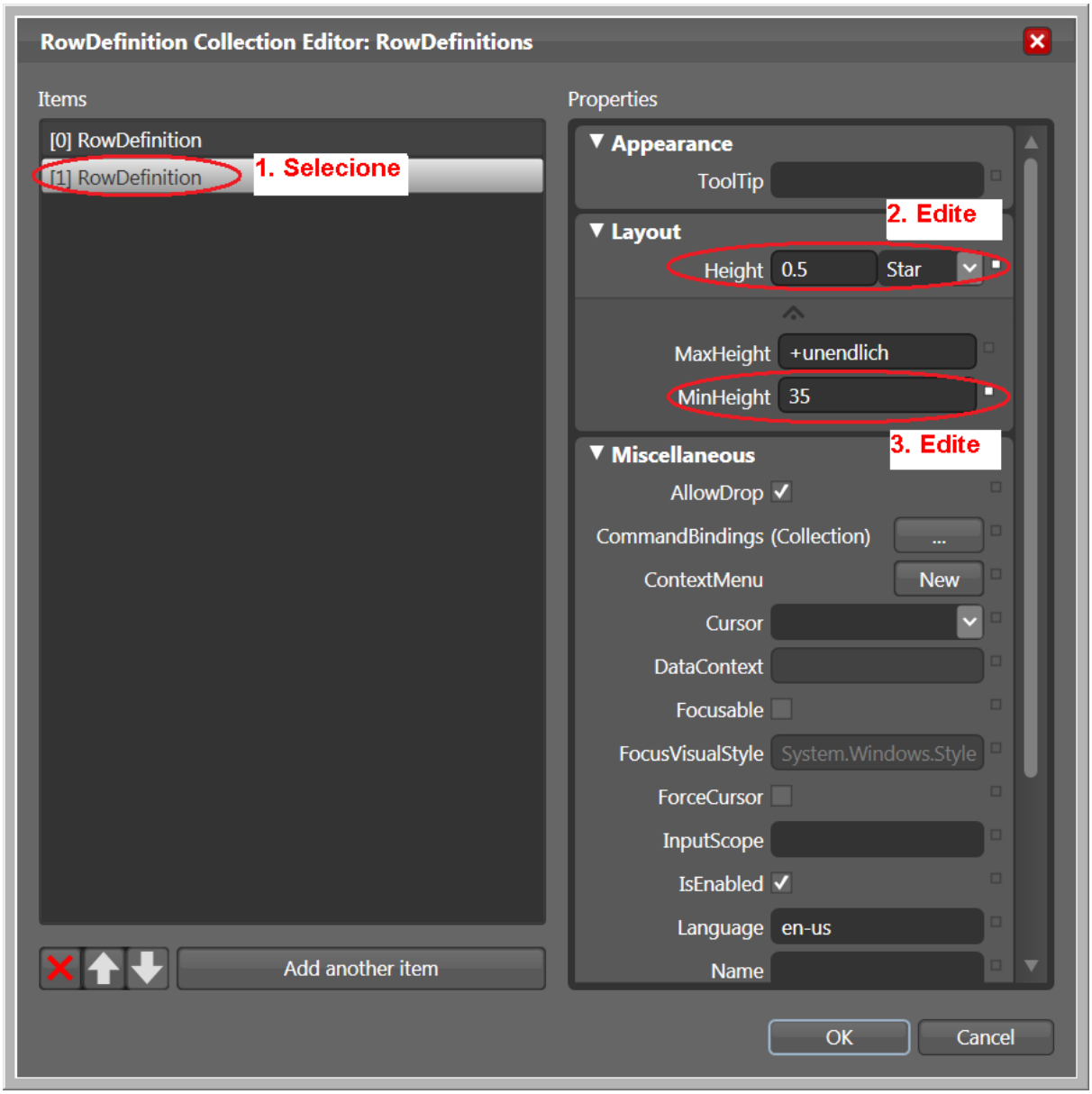


Figura 30 Editando as propriedades da segunda linha do [Grid].

5. Vincule a propriedade **MaxHeight** da segunda linha para o valor do **[StackPanel]. ActualHeight**. **31** Clique no botão quadrado pequeno ao lado da propriedade **MaxHeight** para abrir o menu contextual, escolha o item do submenu **Data Binding ...**; uma caixa de diálogo é aberta; execute as etapas descritas na figura 31 para fazer a ligação de dados.

Em casos normais, uma caixa delimitadora amarela como

será exibida ao lado do nome da propriedade para indicar que a propriedade é um dado vinculado. No entanto, devido a um erro no **Expression Blend**, talvez você não veja, após a ligação de dados, a propriedade **MaxHeight** do **[1] RowDefinition**. Apenas ignore-a.

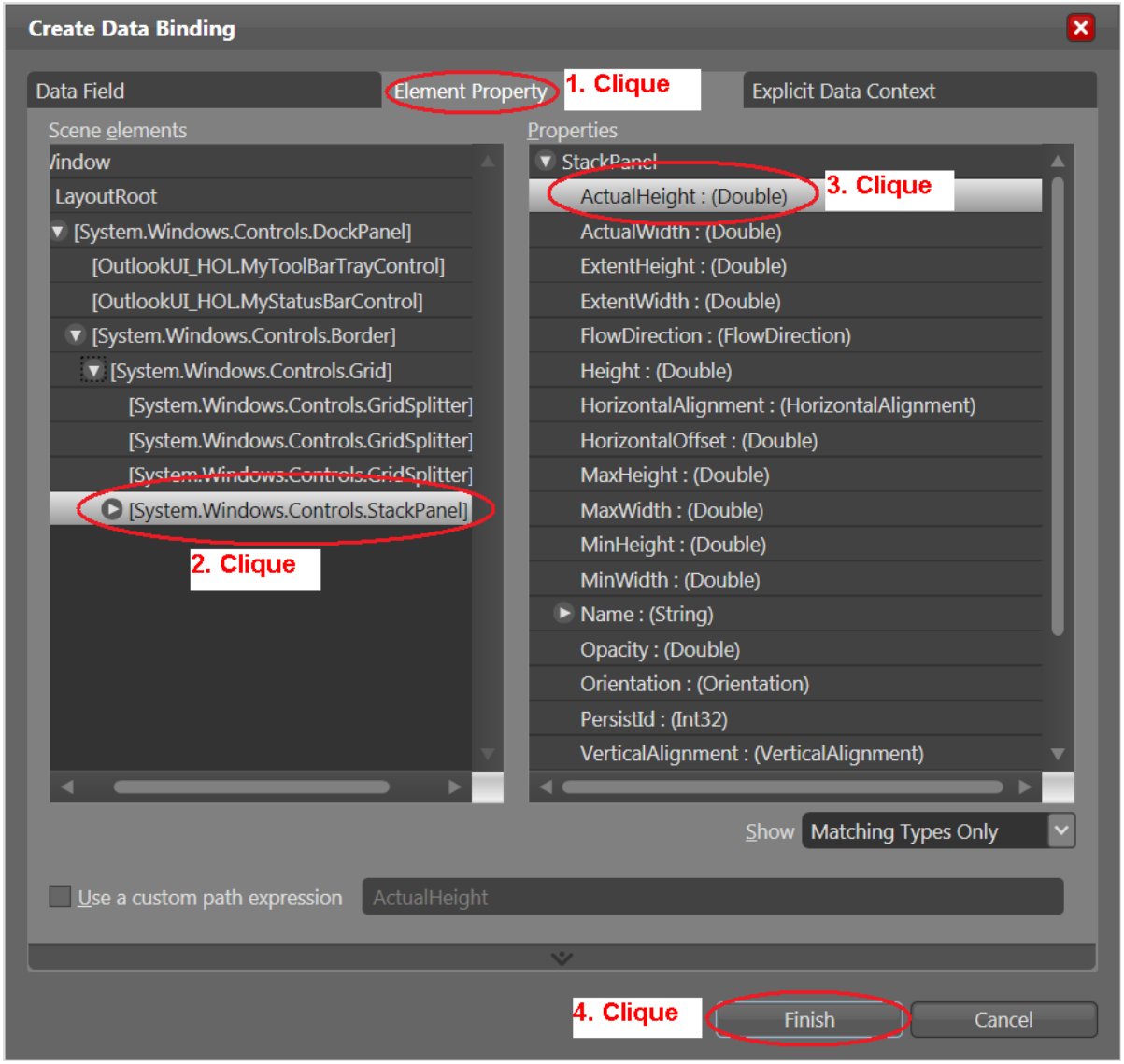


Figura 31 Vinculando a propriedade MaxHeight da segunda linha com o valor da propriedade ActualHeight do [StackPanel].

Clique **OK** para fechar a caixa de diálogo **RowDefinition Collection Editor**.

Após ter configurado a altura da linha do **[Grid]** com êxito, você pode se perguntar: e sobre a largura da coluna? Sim, também precisamos fazer algo com a primeira coluna: Queremos manter **[StackPanel]** visível quando arrastamos o divisor da primeira coluna em direção à esquerda do **[Grid]**. Portanto, vamos definir a propriedade **MinWidth** da primeira coluna como **27** de modo que as imagens sejam sempre visíveis:

1. Selecione o controle **[Grid]**.
2. No **Properties editor**, expanda a categoria **Layout**. Localize a propriedade **ColumnDefinitions (Collection)**, clique no botão ; uma caixa de diálogo, em seguida, é exibida.
3. Defina **MinWidth = 27** e **Name = myMainGridFirstColumn** para **[0] ColumnDefinition**. **32**

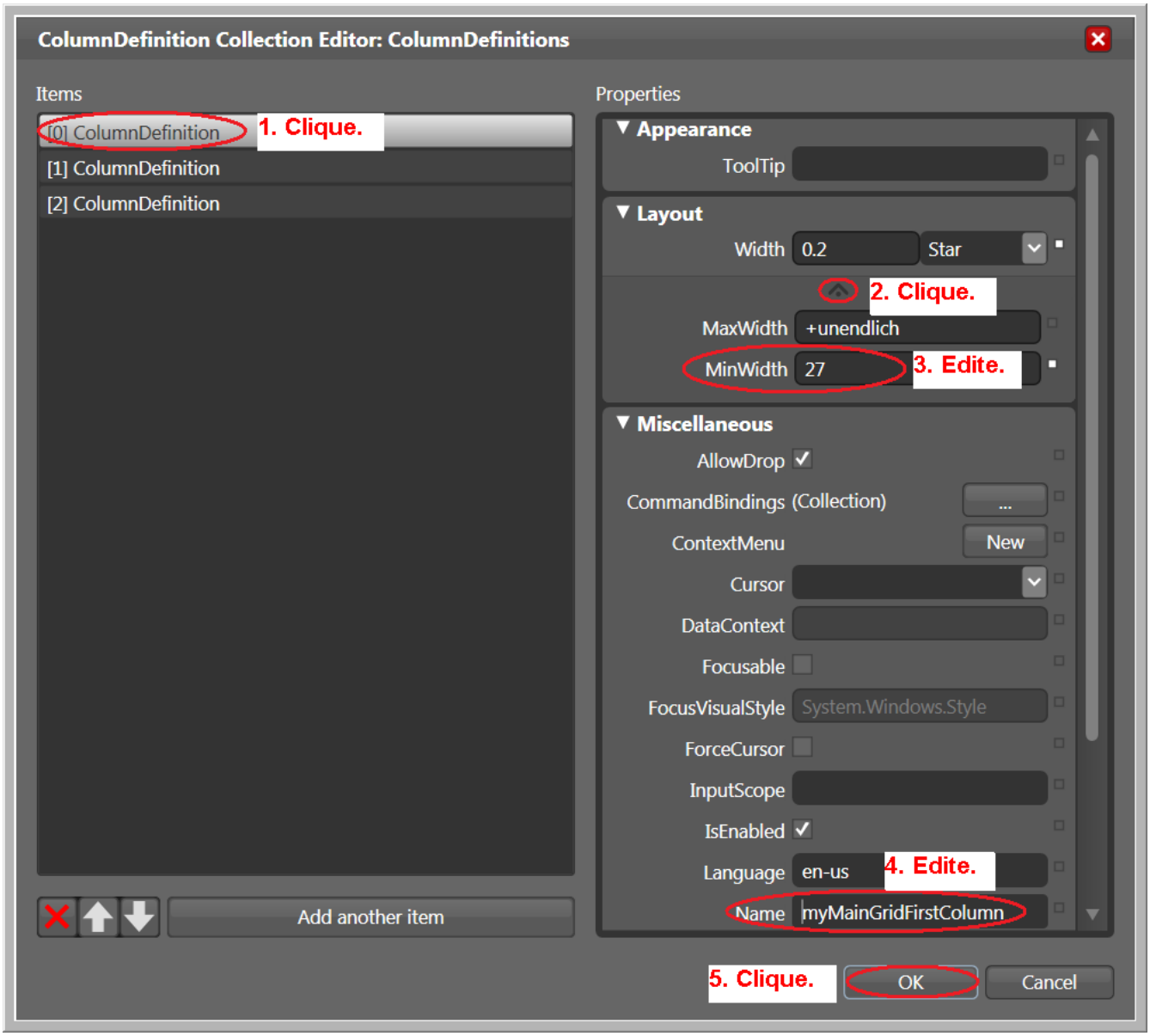


Figura 32 Definindo as propriedades MinWidth e Name da primeira coluna do [Grid].

Crie e execute o aplicativo (**F5**), certifique-se que o aplicativo se comportará como esperado. Convença-se que deve continuar com nosso projeto.

Tarefa 7: Adicionar o MyFoldersExpandersControl ao Upper Left Cell do [Grid]

Os dois expansores para Favorites Folders e Mail Folders estão disponíveis como **Custom Control**, chamado **MyFoldersExpandersControl**.

É hora de aumentar sua confiança nos aplicativos WPF! Tente seu melhor para adicionar um **MyFoldersExpandersControl** no **[Grid]** e configurar suas propriedades da seguinte maneira:

- **Name = myFoldersExpandersControl**

Brushes

- **BorderBrush = MyDarkBlueSolidBrush**

Appearance

- **BorderThickness = 0.5, 0.5, 0.5, 0.5**

Layout

- Width = Auto, Height = Auto
- Row = 0, RowSpan = 1, Column = 0, ColumnSpan = 1
- **HorizontalAlignment = Stretch, VerticalAlignment = Stretch**
- **Margin:** left = 0, right = 0, **top = 32, bottom = 7**

Tarefa 8: Adicionar um ToggleButton para Mostrar/Ocultar o Sidebar

A figura 1 mostra a interface com o usuário do aplicativo cujo **[Grid]** tem três colunas. No entanto, por experiência com o usuário sabemos que a maior parte do tempo nos concentraremos na segunda coluna que mostra todos os itens de uma lista com detalhes, como email do remetente, destinatário, assunto, etc., e a terceira coluna que exibe o conteúdo de um email. Portanto, por que não fazer mais espaço para a segunda e terceira colunas?

Nossa abordagem é criar um controle **ToggleButton** acima dos dois expansores que adicionamos na última tarefa (veja figura 33).

Figura 33 Um controle DockPanel com um TextBlock e um ToggleButton.

Se o **ToggleButton** estiver desmarcado temos, em seguida, o modo de exibição normal com as três colunas no **[Grid]**; se o **ToggleButton** estiver marcado, a primeira coluna é, em seguida, minimizada e é exibido uma barra lateral (sidebar) que contém um botão para o **Navigation Pane** (painel de navegação), e um botão para o **Inbox** é indicado (veja figura 34).

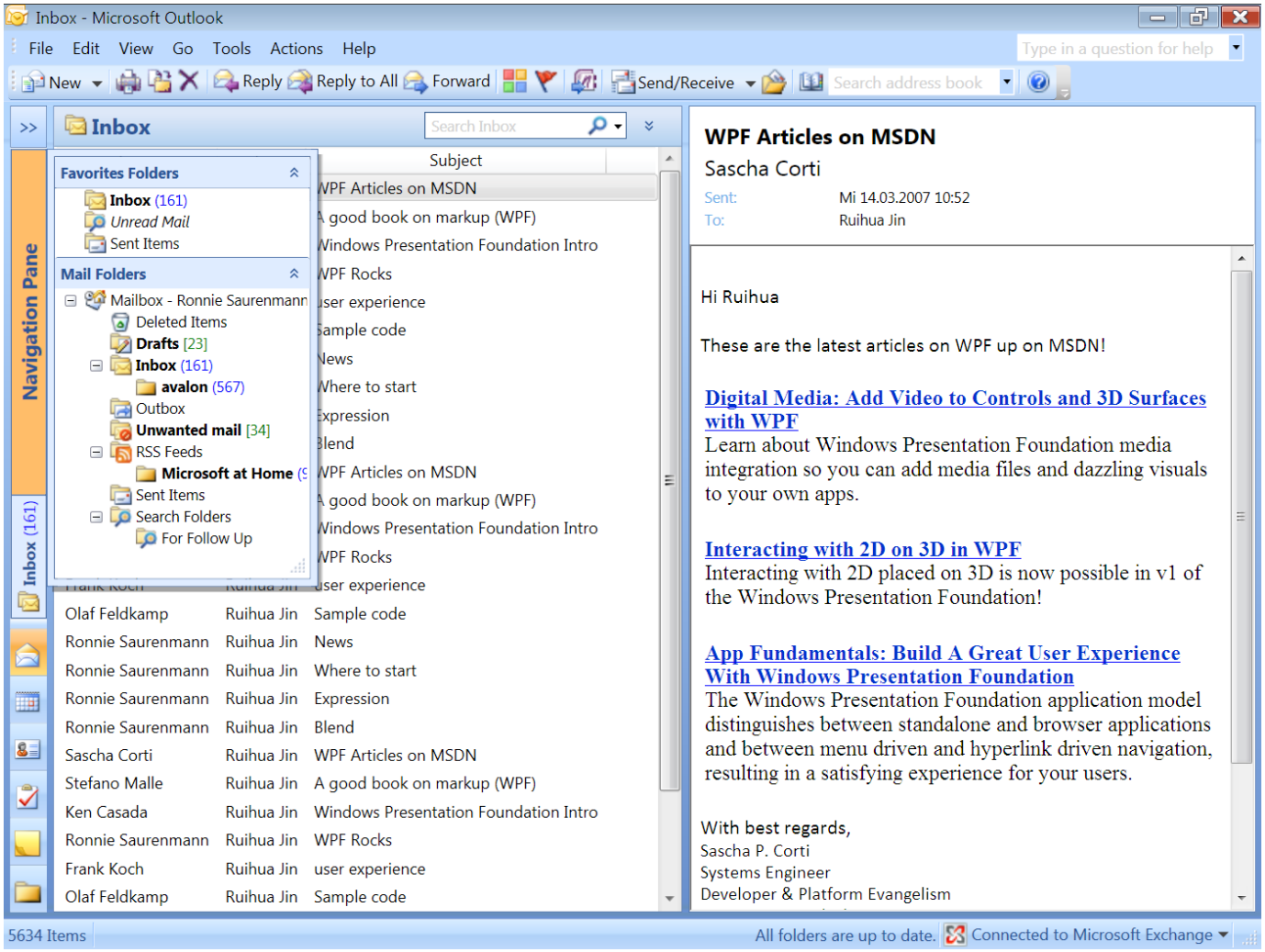

Inbox - Microsoft Outlook
File Edit View Go Tools Actions Help
Type in a question for help
New Reply Reply to All Forward Send/Receive Search address book
Inbox
Search Inbox
Navigation Pane
Inbox (161)
Favorites Folders
Inbox (161)
Unread Mail
Sent Items
Mail Folders
Mailbox - Ronnie Saurenmann
Deleted Items
Drafts [23]
Inbox (161)
avalon (567)
Outbox
Unwanted mail [34]
RSS Feeds
Microsoft at Home
Sent Items
Search Folders
For Follow Up
Subject
WPF Articles on MSDN
A good book on markup (WPF)
Windows Presentation Foundation Intro
WPF Rocks
user experience
Sample code
News
Where to start
Expression
Blend
WPF Articles on MSDN
A good book on markup (WPF)
Windows Presentation Foundation Intro
WPF Rocks
user experience
Olaf Feldkamp Ruihua Jin Sample code
Ronnie Saurenmann Ruihua Jin News
Ronnie Saurenmann Ruihua Jin Where to start
Ronnie Saurenmann Ruihua Jin Expression
Ronnie Saurenmann Ruihua Jin Blend
Sascha Corti Ruihua Jin WPF Articles on MSDN
Stefano Malle Ruihua Jin A good book on markup (WPF)
Ken Casada Ruihua Jin Windows Presentation Foundation Intro
Ronnie Saurenmann Ruihua Jin WPF Rocks
Frank Koch Ruihua Jin user experience
Olaf Feldkamp Ruihua Jin Sample code
WPF Articles on MSDN
Sascha Corti
Sent: Mi 14.03.2007 10:52
To: Ruihua Jin
Hi Ruihua
These are the latest articles on WPF up on MSDN!
Digital Media: Add Video to Controls and 3D Surfaces with WPF
Learn about Windows Presentation Foundation media integration so you can add media files and dazzling visuals to your own apps.
Interacting with 2D on 3D in WPF
Interacting with 2D placed on 3D is now possible in v1 of the Windows Presentation Foundation!
App Fundamentals: Build A Great User Experience With Windows Presentation Foundation
The Windows Presentation Foundation application model distinguishes between standalone and browser applications and between menu driven and hyperlink driven navigation, resulting in a satisfying experience for your users.
With best regards,
Sascha P. Corti
Systems Engineer
Developer & Platform Evangelism
5634 Items
All folders are up to date.
Connected to Microsoft Exchange

Figura 34 OutlookUI com uma barra lateral e um painel de navegação expandido.

Esta parte do trabalho é mais desafiante e excitante do que todos os que nós temos feito até agora; vamos lá!

1. Adicione um controle **DockPanel** no **[Grid]** e nomeie-o como **dp1**.
2. Agrupe o **DockPanel dp1** em um **Border**.
   Em **Objects and Timeline**, clique com o botão direito do mouse em **dp1** para abrir o menu contextual, selecione **Group Into > Border**.

3. Crie um solid brush personalizado para o **BorderBrush** do **[Border]**.

Solid Brush

Um **solid brush** é um brush (pincel) lógico que contém 64 pixels da mesma cor. Após criar o solid brush, o aplicativo pode selecioná-lo em seu contexto de dispositivo e usá-lo para desenhar formas preenchidas.

Com **[Border]** selecionado, clique em **BorderBrush** na categoria **Brushes**, clique então para abrir o editor do solid brush, digite os seguintes valores para vermelho, verde, azul e alfa, para a cor que desejamos:

R = 43, G = 85, B = 151, A = 100%

Esta combinação de valores vermelho, verde, azul e alfa são iguais ao valor Hexadecimal **#FF2B5597**; como alternativa você pode digitar diretamente esse valor na caixa de texto correspondente.

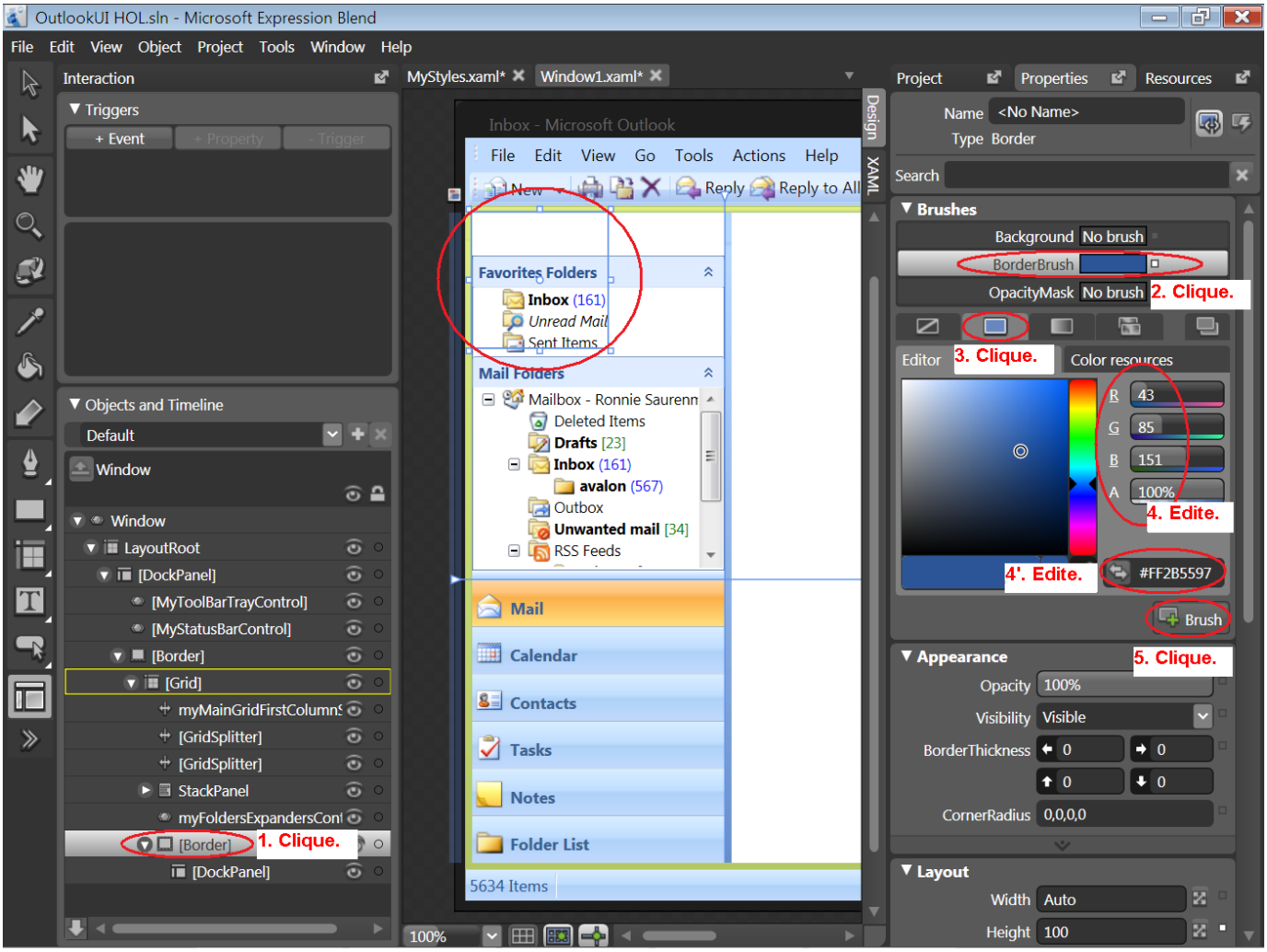
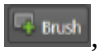


Figura 35 Editando um solid brush personalizado chamado MyBlueBorderSolidBrush.

Em seguida, queremos converter esse brush em um recurso chamado **MyBlueBorderSolidBrush**, essa etapa nos permite reutilizar o brush para propriedades como **Background**, **BorderBrush**, **Foreground** de outros controles. Para alcançar esse objetivo, clique em 

, e uma caixa de diálogo é exibida como é mostrado na figura 36. Digite **MyBlueBorderSolidBrush** na caixa de texto em **Resource name** e escolha **MyStyles.xaml** como o resource dictionary. Clique em **OK**.

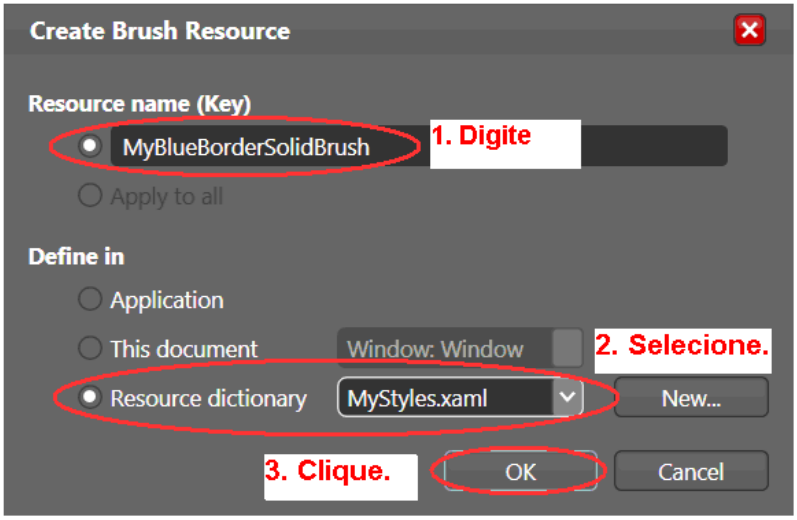


Figura 36 Convertendo o brush MyBlueBorderSolidBrush em um recurso.

4. Crie um linear gradient brush personalizado para a propriedade **Background** do **[Border]**. **37,38,39,40,41**

Linear Gradient Brush

Diferentemente de um solid brush, um **linear gradient brush** usa mais de uma cor. Você define cores diferentes em posições diferentes, e entre duas cores vizinhas, ambas as cores se mesclam para criar uma transição ou efeito de desvanecimento.

1) Com **[Border]** selecionado, clique em **Background** da categoria **Brushes**, e clique em para abrir o gradient brush editor como mostra a figura 37.

Figura 37 O gradient brush editor com seus valores padrão.

Como pode ver, o gradient brush padrão tem cores de mudança gradual, do preto para o branco. Há dois 'pontos' gradientes , um completamente no lado esquerdo e outro no lado direito. Você pode inserir pontos adicionais clicando em qualquer lugar da faixa do gradient editor. Você também pode excluir pontos gradientes, clicando e arrastando-os para fora do gradient editor.

Nosso gradient brush personalizado deve ser semelhante a este:

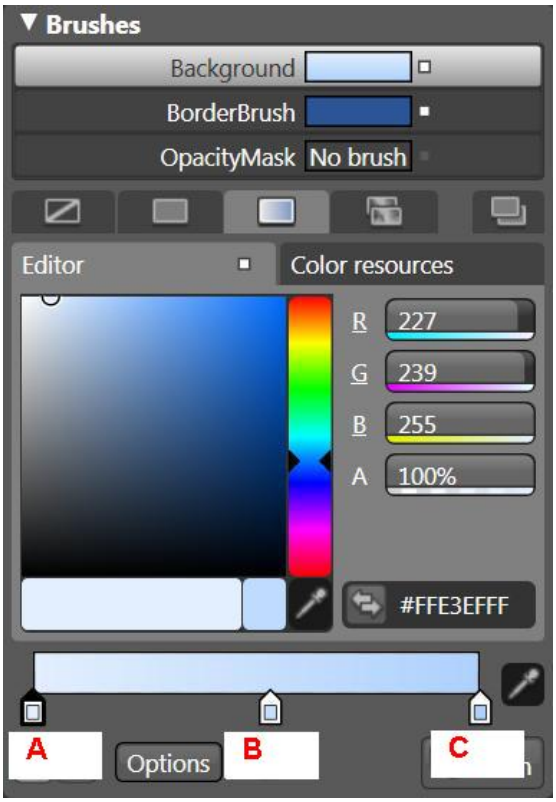


Figura 38 O gradient brush MyBlueBackgroundGradientBrush.

Para criar nosso próprio gradient brush, faça o seguinte:

2) Clique no primeiro ponto gradiente na posição **A** como mostrado na figura 38. Insira os seguintes valores:
   R = 227, G = 239, B = 255, A = 100% ou o **valor Hexadecimal = #FFE3EFFF**
3) Clique na faixa do gradient editor para adicionar um terceiro ponto gradiente e arraste-o na posição **B** como mostrado na figura 38. Insira os seguintes valores:
   R = 199, G = 223, B = 255, A = 100% ou o **valor Hexadecimal = #FFC7DFFF**
4) Clique no terceiro ponto gradiente na posição **C** como mostrado na figura 38, e insira os seguintes valores:
   R = 174, G = 209, B = 255, A = 100% ou o **valor Hexadecimal = #FFAED1FF**
5) Modifique o vetor gradiente:
   Cada gradiente é mapeado por um vetor; você modifica o vetor gradiente para definir os pontos iniciais e finais do gradiente.

   Para obter o vetor gradiente exibido no controle, clique na ferramenta **Brush Transform** no **ToolBox** (veja figura 39).

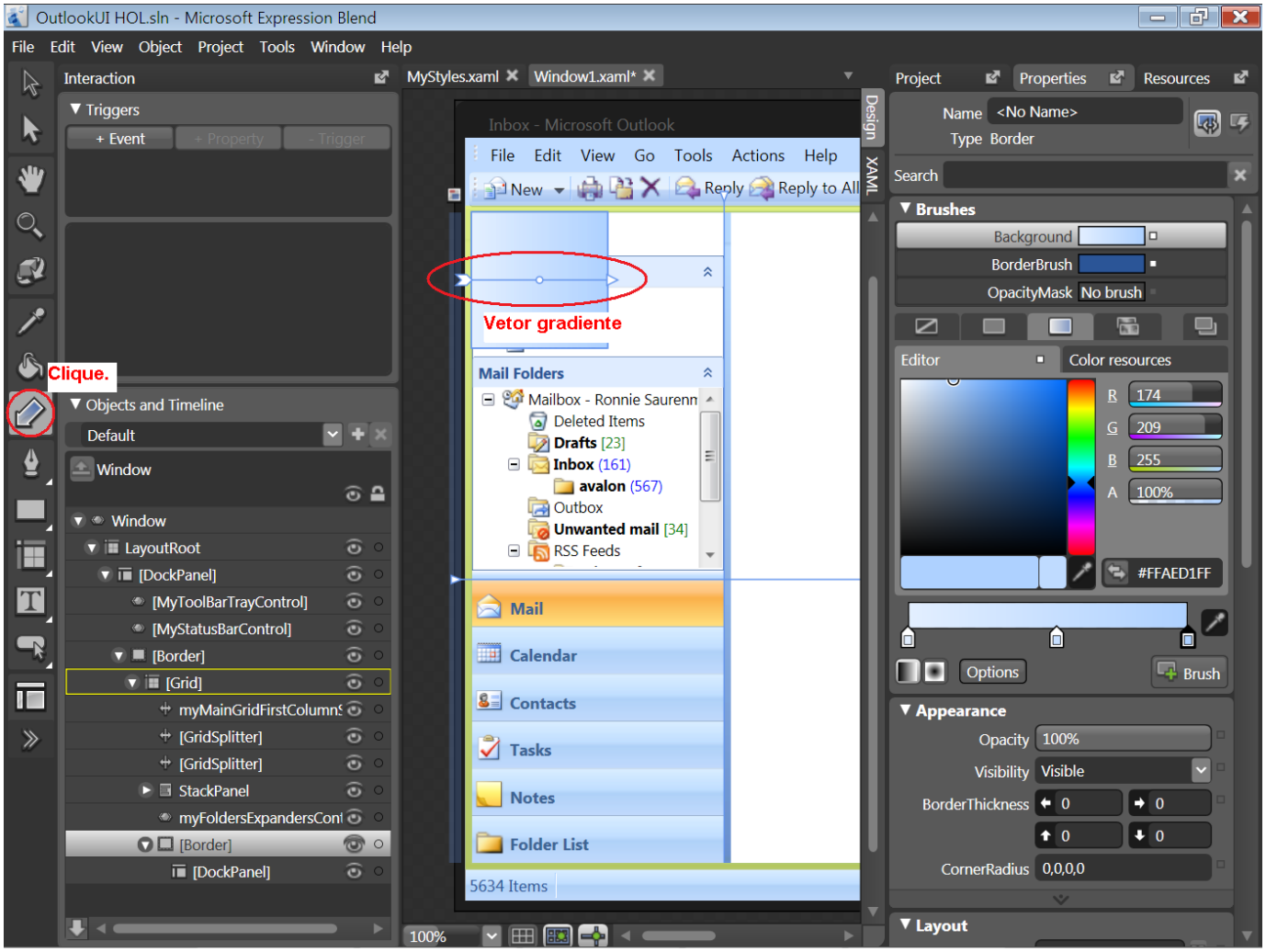


Figura 39 Selecionando a ferramenta Brush Transform para modificar o vetor gradiente.

O vetor gradiente padrão é horizontal, que significa que o gradiente vai da esquerda para direita. Para alterar os pontos iniciais e finais do gradiente, é necessário arrastar a cabeça ou a cauda do vetor. Posicione o mouse à direita da cabeça do vetor até ver um adorne de rotação, mova o cursor do mouse no sentido horário enquanto mantém pressionado o botão esquerdo do mouse e a tecla SHIFT, parando quando o vetor gradiente ficar vertical, conforme é mostrado na figura 40.

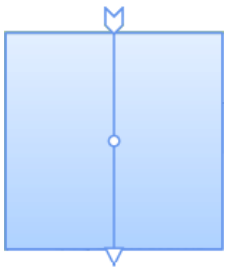


Figura 40 O vetor gradiente foi girado para uma posição vertical.

6) Para converter este brush para um recurso, para reutilização posterior, clique Brush para abrir a caixa de diálogo e insira **MyBlueBackgroundGradientBrush** como nome do recurso e selecione **MyStyles.xaml** como resource dictionary (veja figura 41).

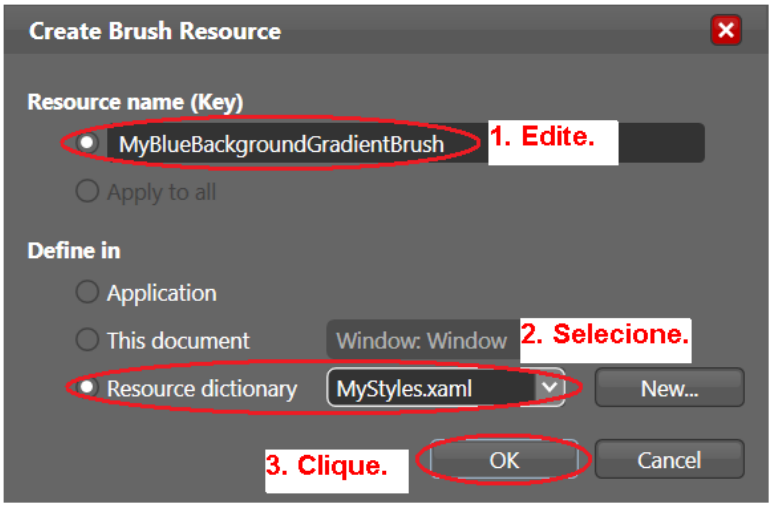


Figura 41 Convertendo o gradient brush em um recurso.

5. Configure outras propriedades do **[Border]** da seguinte maneira:
   Appearance
   - **BorderThickness = 0.5, 0.5, 0.5, 0.5**

   Layout
   - **Width = Auto, Height = 31**
   - **HorizontalAlignment = Stretch, VerticalAlignment = Top**
   - **Margin = 0, 0, 0, 0**
6. Com **dp1** selecionado, configure as propriedades do **dp1** da seguinte maneira:
   Layout
   - **Width = Auto, Height = Auto**
   - HorizontalAlignment = Stretch, VerticalAlignment = Stretch
   - Margin = 0, 0, 0, 0
7. Com **dp1** ativado, adicione um controle **TextBlock** no **dp1** com as seguintes propriedades:
   Brushes
   - **Foreground = MyDarkBlueSolidBrush**

   Layout
   - Width = Auto, Height = Auto
   - Dock = Left
   - HorizontalAlignment = Stretch, **VerticalAlignment = Center**
   - **Margin: left = 4,** right = 0, top = 0, bottom = 0

   Common Properties
   - **Text = Mail**

   Text
   - **FontSize = 16, FontWeight = Bold**
   - **TextWrapping = NoWrap**
8. Adicione um controle **ToggleButton** no **dp1** com as seguintes propriedades:
   - **Name = myMainGridToggleButton**

   Brushes
   - **Background = No brush**
   - **BorderBrush = No brush**

   Layout
   - **Width = 26,** Height = Auto
   - **Dock = Right**
   - HorizontalAlignment = Stretch, **VerticalAlignment = Center**
   - Margin = 0, 0, 0, 0

Agora nós temos um **ToggleButton**, mas não está com a aparência que nós queremos, nós queremos « em vez de um botão retangular. Uma grande vantagem do WPF é que os controles não são forçados a ter uma aparência predefinida ou um comportamento que estamos habituados, nós podemos realmente personalizar e controlar estilos do nosso gosto e executar à maneira que desejamos. Por exemplo, um controle **Button** pode ser retangular, pode ser redonda ou ainda pode ter a forma de um avião.

Controls, Templates de controle e Styles

Os **controls** (controles) são os componentes básicos da interface com o usuário em aplicativos, manipulando a maioria das interações com o usuário com o aplicativo. Windows Presentation Foundation contém um número de templates de controle padrão que definem a aparência dos controles de janelas em diferentes temas. Você pode alterar a estrutura e aparência de um controle, editando o modelo de controle, no controle.

Cada controle consiste em duas partes principais — um template e um style.

Um **template** (modelo) é o elemento modelo que é usado para criar um controle. Você pode editar um template de controle e reorganizar, adicionar ou excluir os elementos (ou partes) dentro do controle.

Um **style** (estilo) permite que você defina como os atributos padrão são definidos para um controle. Esses incluem estados, como a aparência ao pressionar um botão, bem como as cores padrão do botão. Styles são essencialmente "pacotes de propriedades" que, em muitos casos, igualmente designam qual template o controle usará.

9. Faça o seguinte para definir nossa própria seta-dupla-ToggleButton:

    1) Entre no **Control Template Editor** no **myMainGridToggleButton**. 42 43
    Em **Objects and Timeline**, clique com o botão direito do mouse no **myMainGridToggleButton** para abrir o menu contextual, selecione o item do submenu **Edit Control Parts (Template) > Edit a Copy ...** (você não pode modificar os styles e os templates do sistema, então faça cópias deles).

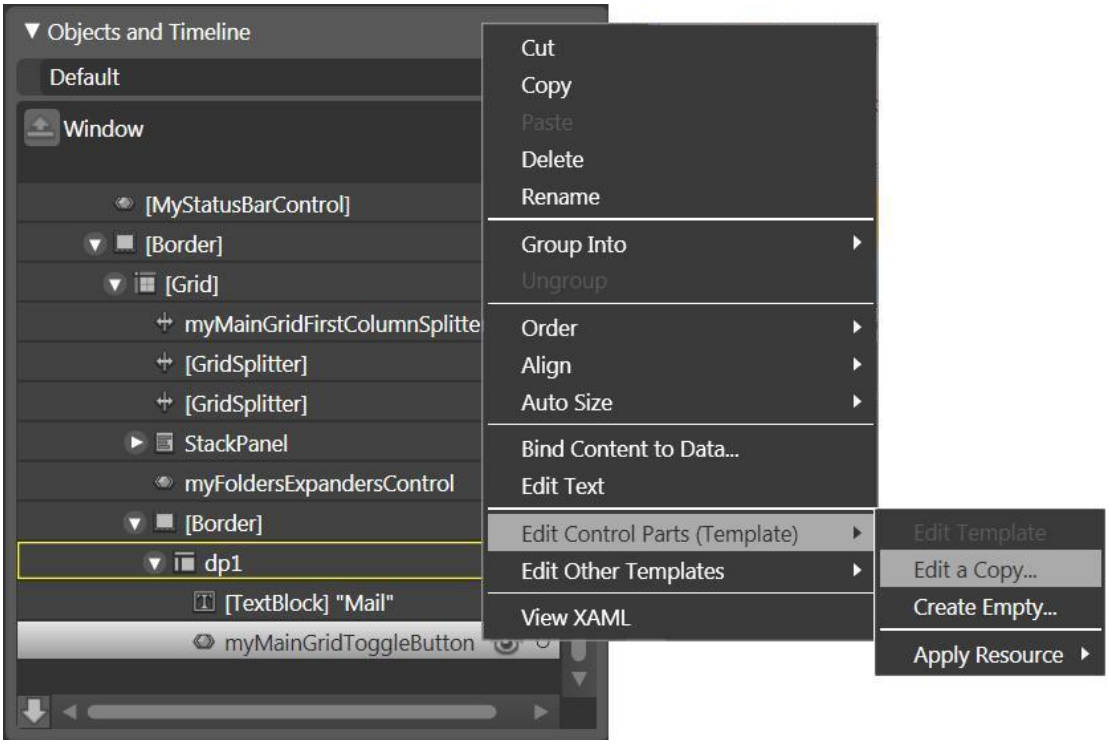


Figura 42 Inserindo o Control Template Editor no myMainGridToggleButton.

Em seguida, uma caixa de diálogo é aberta; aqui você especifica o nome do style como **MyDoubleArrowToggleButtonStyle**, e seleciona **MyStyles.xaml** como um resource dictionary. Clique em **OK**.

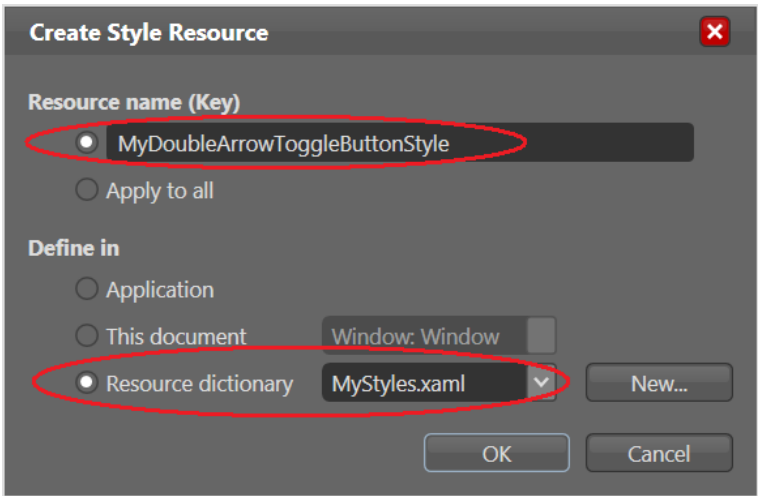
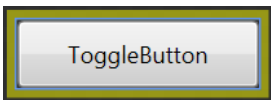


Figura 43 Nomeando o style e selecionando um resource dictionary.

Como pode ver, o modelo **ToggleButton** 

tem um **Chrome** que contém um controle **[ContentPresenter]** (veja figura 44).

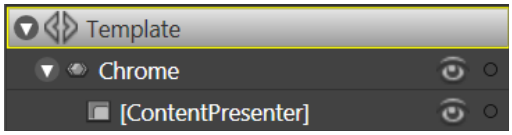


Figura 44 Estrutura hierárquica de controle do controle original ToggleButton.

2) Para excluir **Chrome**, clique com o botão direito do mouse no **Chrome**, para abrir o menu contextual, e selecione **Delete**.
3) Com **Template** ativado, adicione um controle **Grid** no **Template** e configure suas propriedades da seguinte maneira:
   Brushes

   - **Background = um solid color brush arbitrário com Alpha = 0%**
     Na categoria **Brushes**, clique em **Background** e clique para inserir o **Solid color brush editor**. Escolha uma cor arbitrária e defina **A=0%**. Um valor **alfa** de **zero** representa total transparência.

Você pode querer saber por que nós executamos esta etapa para conseguir um efeito de nenhum efeito. Como você verá depois que adicionar os dois controles **Path**s no **[Grid]**, se o controle **[Grid]** não tiver uma cor **Background**, o usuário deve clicar exatamente no **Path** para marcar ou desmarcar o **ToggleButton**; o que é inconveniente. Uma cor transparente como cor background do **[Grid]** resolve esse problema.

Layout

- **Width = 26,** Height = Auto

4) Com **[Grid]** ativado, desenhe um path no **[Grid]** no artboard. **45**

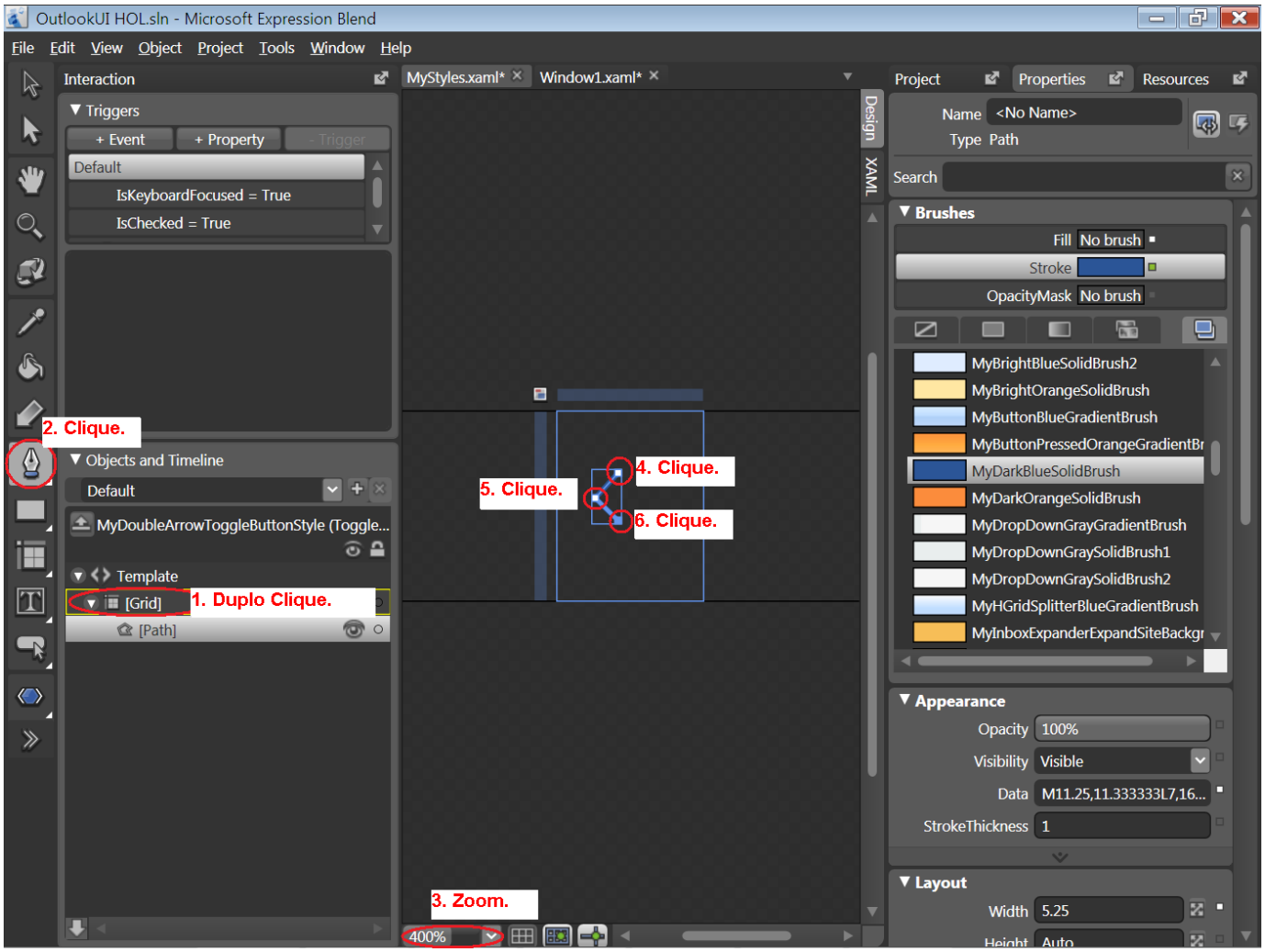


Figura 45 Desenhando um path dentro do [Grid].

5) Configure as propriedades do controle **[Path]** da seguinte maneira:
Brushes

- **Fill = No brush**
- **Stroke = MyDarkBlueSolidBrush**

6) Crie um segundo path no **[Grid]**. 46
Em **Objects and Timeline**, clique com o botão direito do mouse no **[Path]** para abrir o menu contextual, selecione **Copy**. Clique com o botão direito do mouse no **[Grid]** para abrir o menu contextual, selecione **Paste**.
Com o segundo **[Path]** selecionado, arraste-o para a direita conforme descrito na figura 46.

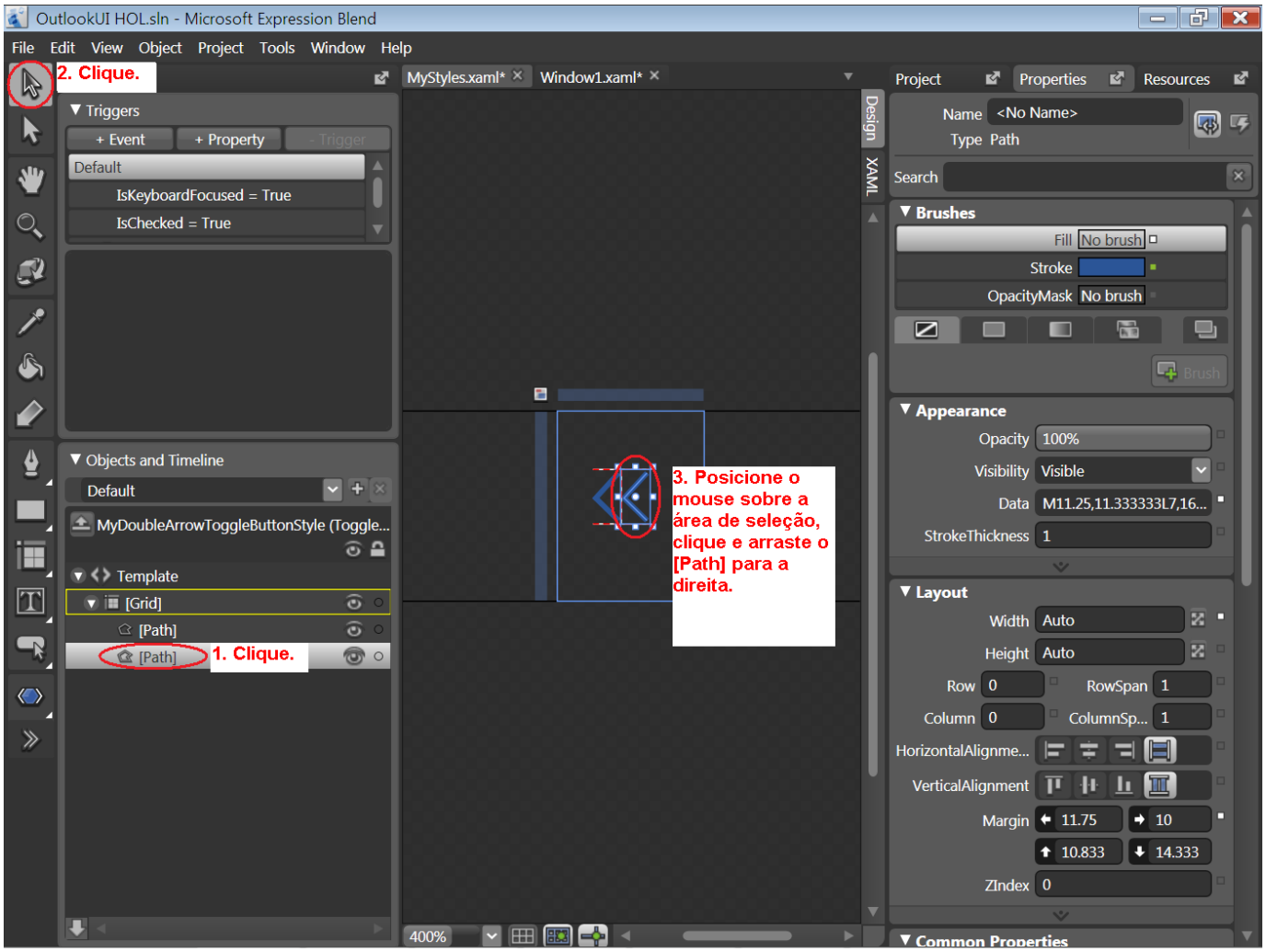


Figura 46 Movendo o segundo path para a direita.

7) Adicione e edite o property trigger **IsChecked=True**: 47 48
Como desejamos nossa seta-dupla-ToggleButton para apontar para a esquerda ou direita, dependendo se o usuário marcá-lo ou desmarcá-lo, precisamos adicionar um property trigger que é ativado quando a propriedade **IsChecked** fica **True**.

Property Trigger

Property trigger é o mecanismo pelo qual uma alteração em uma propriedade dispara um evento *instantâneo* ou *animado* que altera outra propriedade.

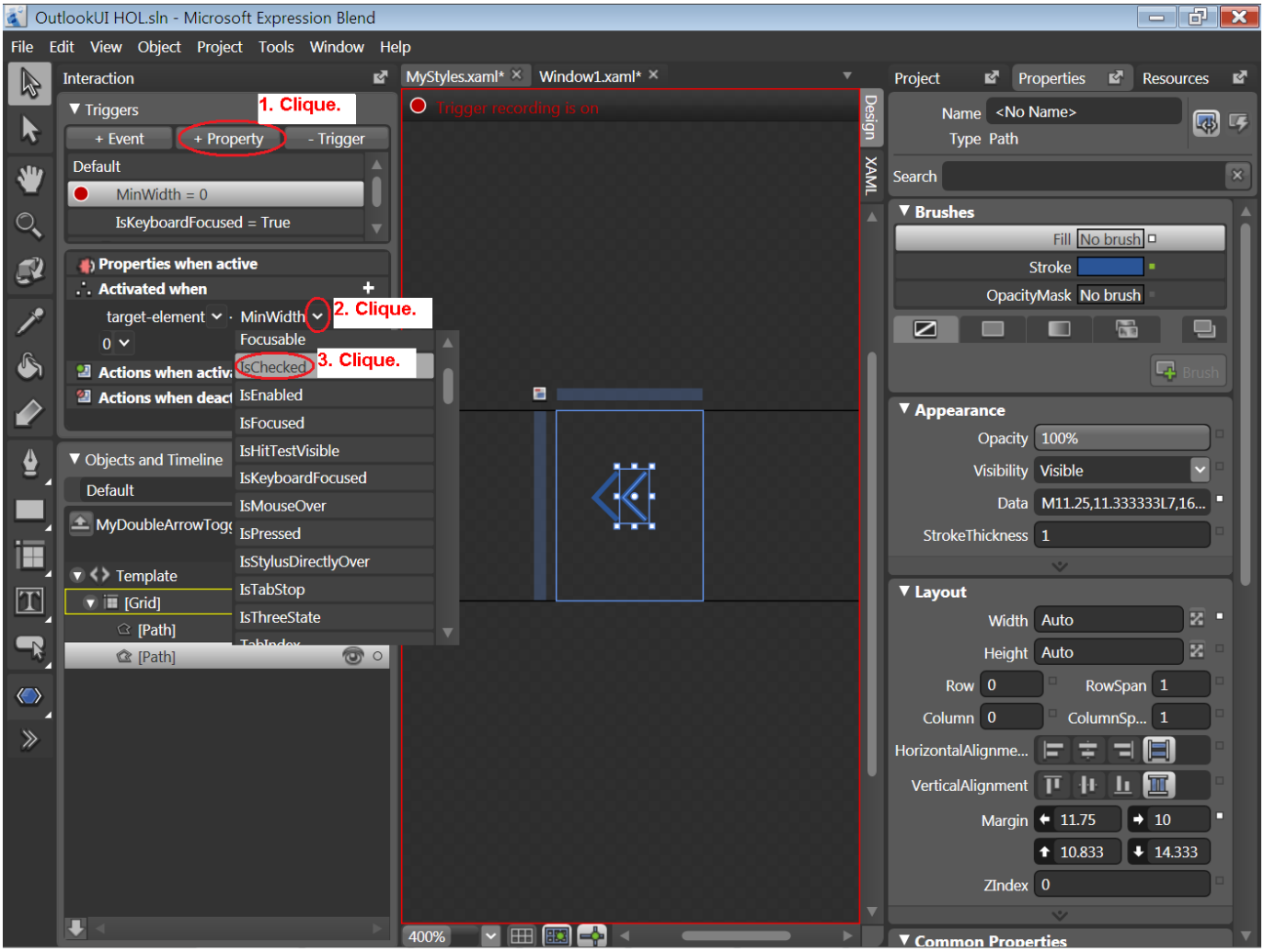
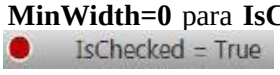
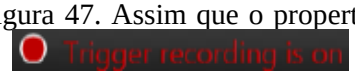


Figura 47 Adicionando o property trigger IsChecked=True.

MinWidth=0 é o padrão de property trigger; altere **MinWidth=0** para **IsChecked=True** conforme descrito na figura 47. Assim que o property trigger 

for adicionado à lista de disparadores, 

é exibido na tela, que significa que todas as alterações feitas a partir de agora em diante são registradas no property trigger, eles terão efeito quando a propriedade relacionada obtém o valor especificado. No nosso caso, quando **ToggleButton.IsChecked = True**, a seta-dupla aponta para a direção oposta. As alterações de propriedades, para ambos os controles **[Path]**, são:

Transform

- Na subcategoria **RenderTransform**, **Angle=180**.

Consulte a figura 48 para configurar as propriedades de ambos os controles **[Path]**.

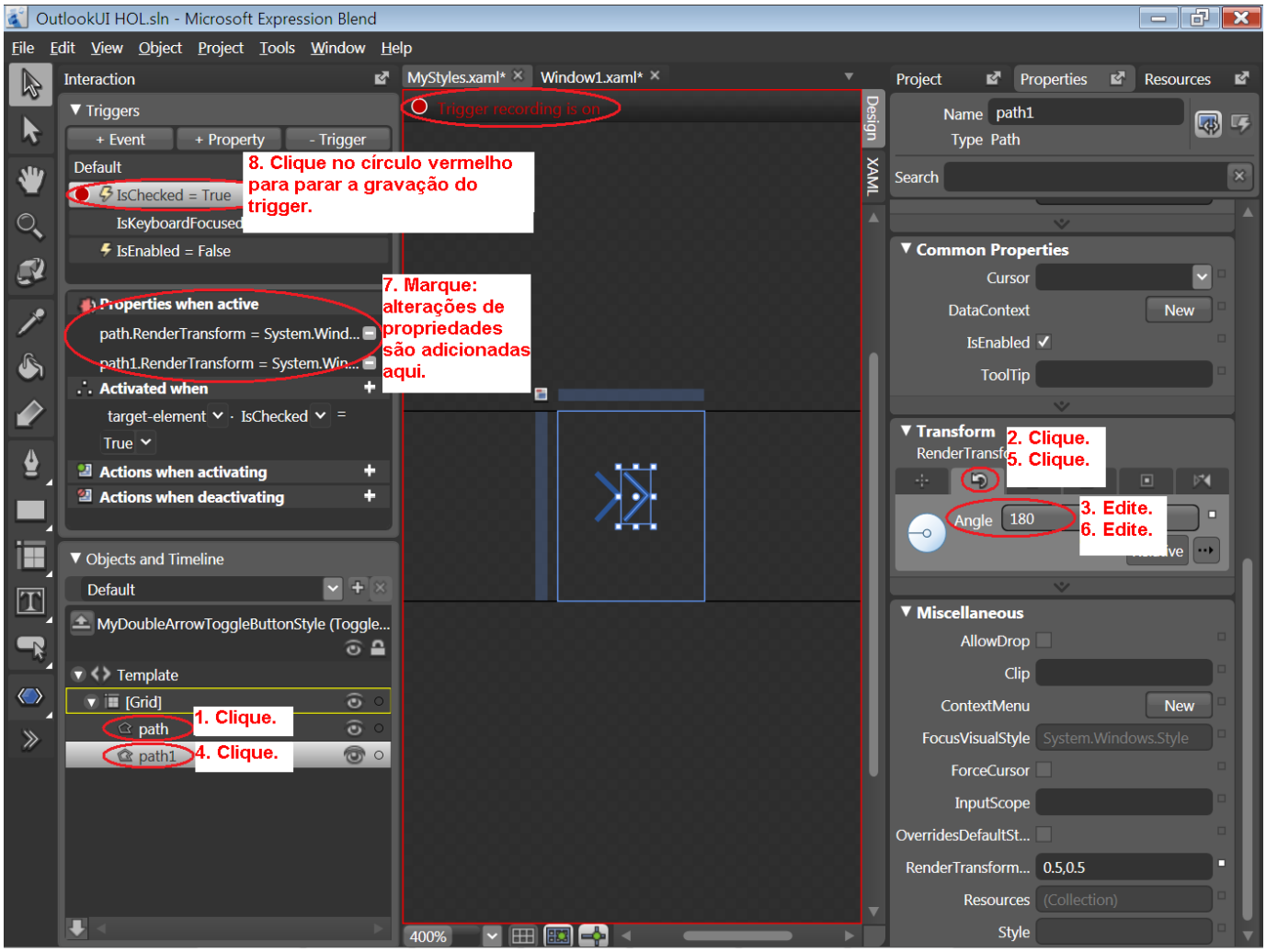


Figura 48 Editando o property trigger IsChecked=True.

8) Em **Objects and Timeline**, clique em (espaço acima) para sair do Control Template Editor.

Crie e execute o aplicativo (**F5**), certifique-se que o **ToggleButton** funcione conforme o esperado.

A próxima etapa é adicionar um sidebar. Execute as seguintes etapas:

1. Com **[Grid]** ativado, adicione um controle personalizado **MySidebarControl** no **[Grid]**.
2. Configure as propriedades do **[MySidebarControl]** da seguinte maneira:
   - **Name = mySidebarControl**

Brushes

   - **BorderBrush = MyDarkBlueSolidBrush**

Appearance

   - **BorderThickness = 0.5, 0.5, 0.5, 0.5**

Layout

   - Width = Auto, **Height = 27**
   - **HorizontalAlignment = Left, VerticalAlignment = Stretch**
   - **Margin:** left = 0, right = 0, **top = 32, bottom = 7**

Transform

Expanda a categoria **Transform**, sob a subcategoria **LayoutTransform**. 49

- **Angle = -90**

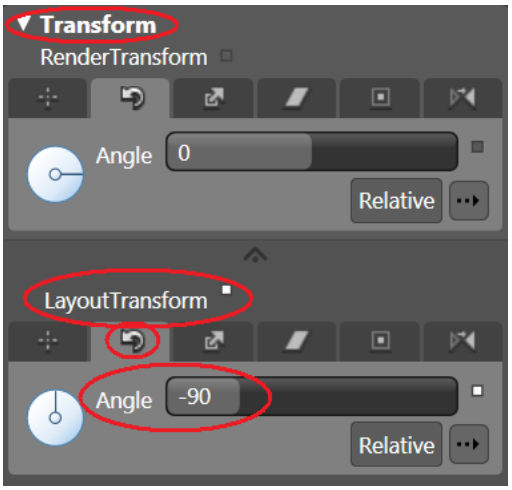


Figura 49 Editando a propriedade Angle do LayoutTransform.

Appearance

- **Visibility = Hidden**

Conforme descrito acima, desejamos obter a primeira coluna do grid minimizada e o sidebar exibido quando o **myMainGridToggleButton** for marcado. Para obter esse efeito, precisamos escrever um event handler (manipulador de eventos) para o evento **myMainGridToggleButton.Checked**. Volte ao modo de exibição padrão, e escreva um event handler para o evento **myMainGridToggleButton.Unchecked**.

Event Handling (manipulação do evento)

Event handling é o mecanismo que permite interações flexíveis entre componentes de programas. Para reagir a um determinado evento, você deve registrar um event handler para o evento; assim que esse evento for disparado, seu event handler é chamado. Eventos típicos são eventos GUI (Graphical User Interface – Interface Gráfica com o Usuário), como cliques do mouse, pressionar chaves, etc.

1. Escreva um event handler para o evento **myMainGridToggleButton.Checked**. 50

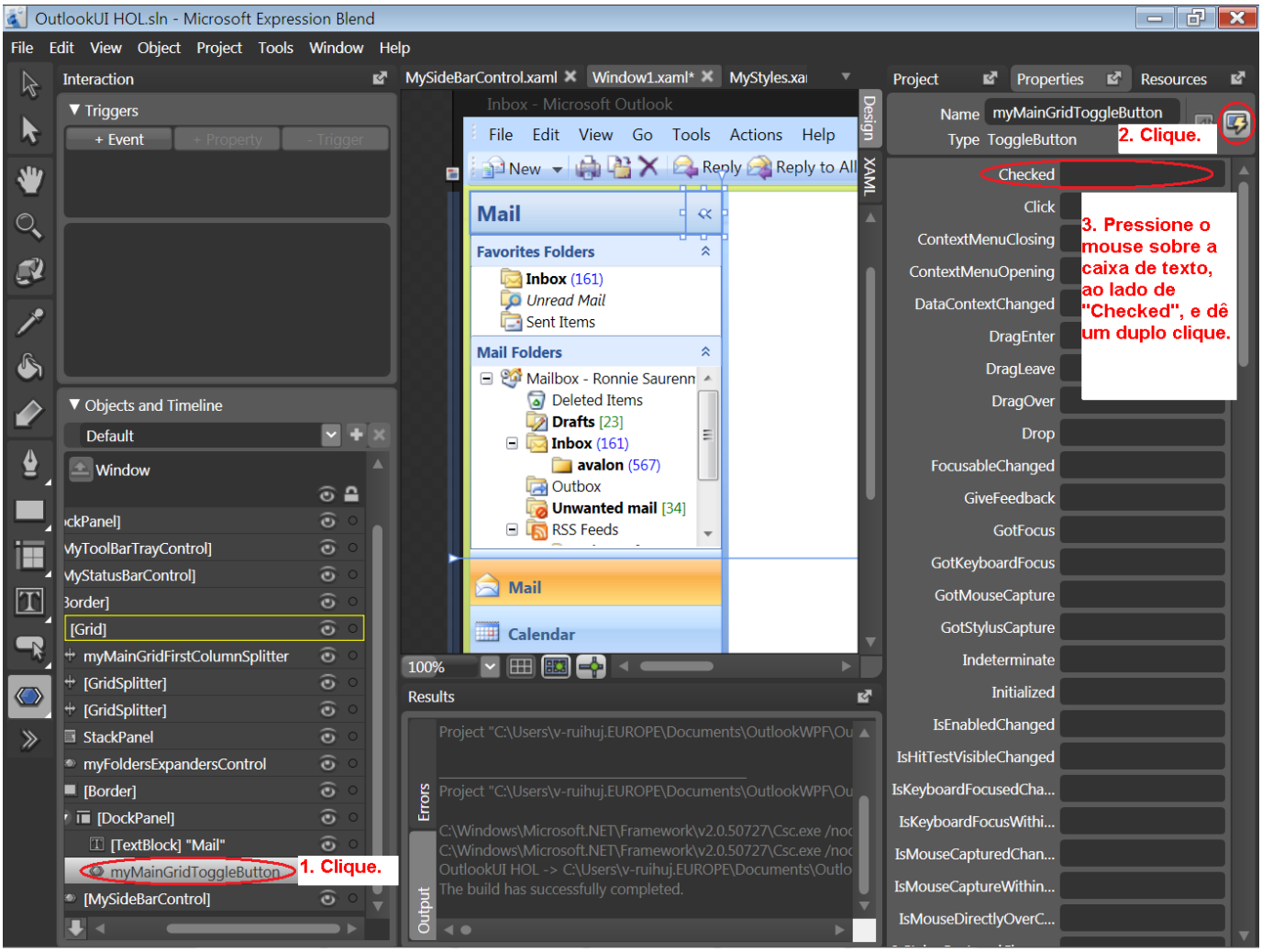


Figura 50 Adicionando um event handler ao evento myMainGridToggleButton.Checked.

Após a etapa 3, descrita na figura 50, o Expression Blend nomeia automaticamente o event handler como **myMainGridToggleButton_Checked**.

Se você tem o Visual Studio instalado, o Visual Studio será inicializado, e a assinatura do método é adicionada automaticamente no **Window1.xaml.cs**; se não, escolha seu editor favorito para editar o event handler.

No arquivo **Window1.xaml.cs**, a classe **Window1** é definida no namespace **OutlookUI_HOL**. Adicione o código a seguir (negrito) para o método **myMainGridToggleButton_Checked** na classe **Window1**:

basic8.txt

```
private GridLength gridLength;
private void myMainGridToggleButton_Checked(object sender, RoutedEventArgs e)
{
   /* store the current width of the first grid column */
   if (myMainGridFirstColumn.ActualWidth > myMainGridFirstColumn.MinWidth)
   gridLength = myMainGridFirstColumn.Width;
   /* set the first grid column width to its minimum width */
   myMainGridFirstColumn.Width = new GridLength(myMainGridFirstColumn.MinWidth);
   /* hide the two expanders for Favorites Folders and Mail Folders */
   myFoldersExpandersControl.Visibility = Visibility.Hidden;
   /* display mySidebarControl */
   mySidebarControl.Visibility = Visibility.Visible;
}
```

2. Escrever um event handler para o evento **myMainGridToggleButton.Unchecked**.

basic8.txt

```
private void myMainGridToggleButton_Unchecked(object sender, RoutedEventArgs e)
{
  if (!myMainGridFirstColumnSplitter.IsDragging)
      myMainGridFirstColumn.Width = gridLength; // restore the first column width
  myFoldersExpandersControl.Visibility = Visibility.Visible;
  mySidebarControl.Visibility = Visibility.Hidden;
}
```

3. Além do **myMainGridToggleButton**, o primeiro **myMainGridFirstColumnSplitter** do divisor da coluna do grid também altera a largura da primeira coluna quando você o arrasta, que é por isso que precisamos escrever um event handler para o evento **myMainGridFirstColumnSplitter.DragDelta**:

basic8.txt

```
private void myMainGridFirstColumnSplitter_DragDelta(
          object sender, System.Windows.Controls.Primitives.DragDeltaEventArgs e)
{
  /** trigger the myMainGridToggleButton.Checked event when
      * the first grid column width reaches its minimum */
  if (myMainGridFirstColumn.ActualWidth <= myMainGridFirstColumn.MinWidth)
      myMainGridToggleButton.IsChecked = true;
  /** trigger the myMainGridToggleButton.Unchecked event when
      * the first grid column width exceeds its minimum */
  else
      myMainGridToggleButton.IsChecked = false;
}
```

Crie e execute o aplicativo (**F5**). Arraste o primeiro divisor da coluna para a esquerda e direita; marque o **ToggleButton**. O sidebar aparecerá e começará oculto conforme o esperado. Você pode reclamar que o texto **Mail** se mantém grudado na posição; não se preocupe, há uma maneira fácil para resolver esse problema, apenas faça o seguinte:

4. Defina a ordem de **myMainGridToggleButton**. **51**
Clique com o botão direito do mouse no **myMainGridToggleButton** para abrir o menu contextual, selecione **Order > Send to Back**. Você pode ver que **myMainGridToggleButton** é movido acima do **[TextBlock] " Mail "**, isso significa que o **TextBlock " Mail "** se torna o último da herança do **DockPanel dp1**. Com **dp1.LastChildFill**, defina para **True**; **myMainGridToggleButton** é posicionado em primeiro, e o **TextBlock** preenche a área restante.

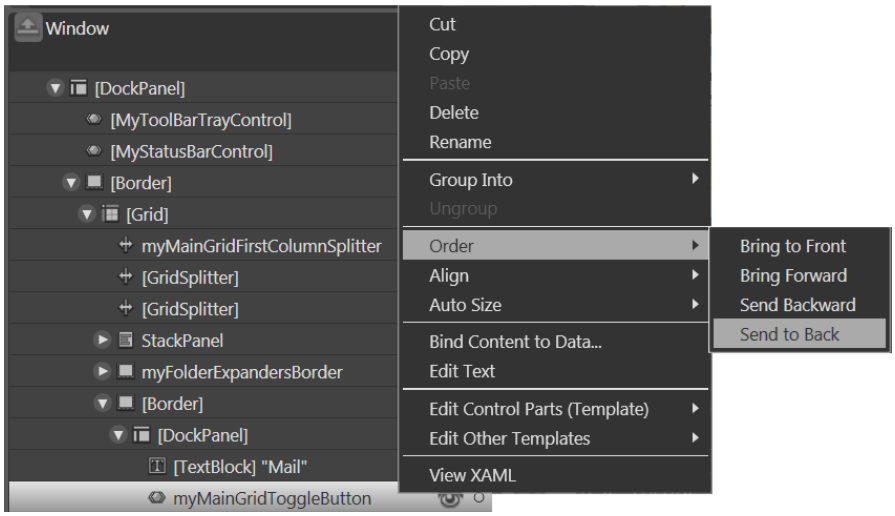


Figura 51 Definindo a ordem do myMainGridToggleButton.

Tarefa 9: Adicionar um MyInboxExpanderControl à Segunda Coluna do [Grid]

A segunda coluna do nosso recipiente de layout **[Grid]** tem um controle **Expander** e um controle **ListView** que mostra todos os itens do Inbox.

1. Com **[Grid]** ativado, adicione um controle **DockPanel** no **[Grid]** e nomeie-o como **dp2**.

Como nós alinhamos dois controles verticalmente em uma direção, um **StackPanel** como controle pai seria mais adequado. No entanto, a prática revela que um **StackPanel** não trata bem um controle **ScrollViewer** contido nele, o scrollbar não é exibido mesmo quando o conteúdo de um controle **ScrollViewer** excede o intervalo do controle **StackPanel**. Um **DockPanel** preferivelmente não sofre deste problema.

2. Agrupe o **DockPanel dp2** em um **Border**.
3. Configure as propriedades do **[Border]** da seguinte maneira:
   Brushes
   - **BorderBrush = MyDarkBlueSolidBrush**

   Appearance
   - **BorderThickness = 0.5, 0.5, 0.5, 0.5**

   Layout
   - **Width = Auto, Height = Auto**
   - Row = 0, **RowSpan = 2, Column = 1,** ColumnSpan = 1
   - **HorizontalAlignment = Stretch, VerticalAlignment = Stretch**
   - **Margin: left = 5, right = 5,** top = 0, bottom = 0
4. Configure as propriedades de **dp2** da seguinte maneira:
   Layout
   - **Width = Auto, Height = Auto**
   - HorizontalAlignment = Stretch, VerticalAlignment = Stretch
   - **LastChildFill = True**
   - Margin = 0, 0, 0, 0
5. Com **dp2** ativado, adicione um controle personalizado chamado **MyInboxExpanderControl** no **dp2**, e configure suas propriedades da seguinte maneira:
   Layout
   - **Dock = Top**
   - HorizontalAlignment = Stretch, **VerticalAlignment = Top**

Tarefa 10: Adicionar um controle ListView no Display Mail

Nesta tarefa, você aprenderá a como vincular um controle a um XML data source externo.

1. Com a **DockPanel dp2** ativado, adicione um controle **ListView** no **dp2** e configure suas propriedades da seguinte maneira:
   Brushes
   - **BorderBrush = No brush**
   - **Foreground = Black** (no editor solid color brush, defina **R = 0, G = 0, B = 0, A = 100 %**).

   Appearance
   - **BorderThickness = 0, 0, 0, 0**

   Layout
   - **Dock = Top**
   - Margin = 0, 0, 0, 0

   Common Properties
   - IsSynchronizedWithCurrentItem = True (Marcado)
   - **SelectedIndex = 0**

2. Adicione um novo XML data source no Projeto. **52** **53**
   Alterne para a guia **Project**, na seção **Data**, clique no botão **+XML**; uma caixa de diálogo é aberta.

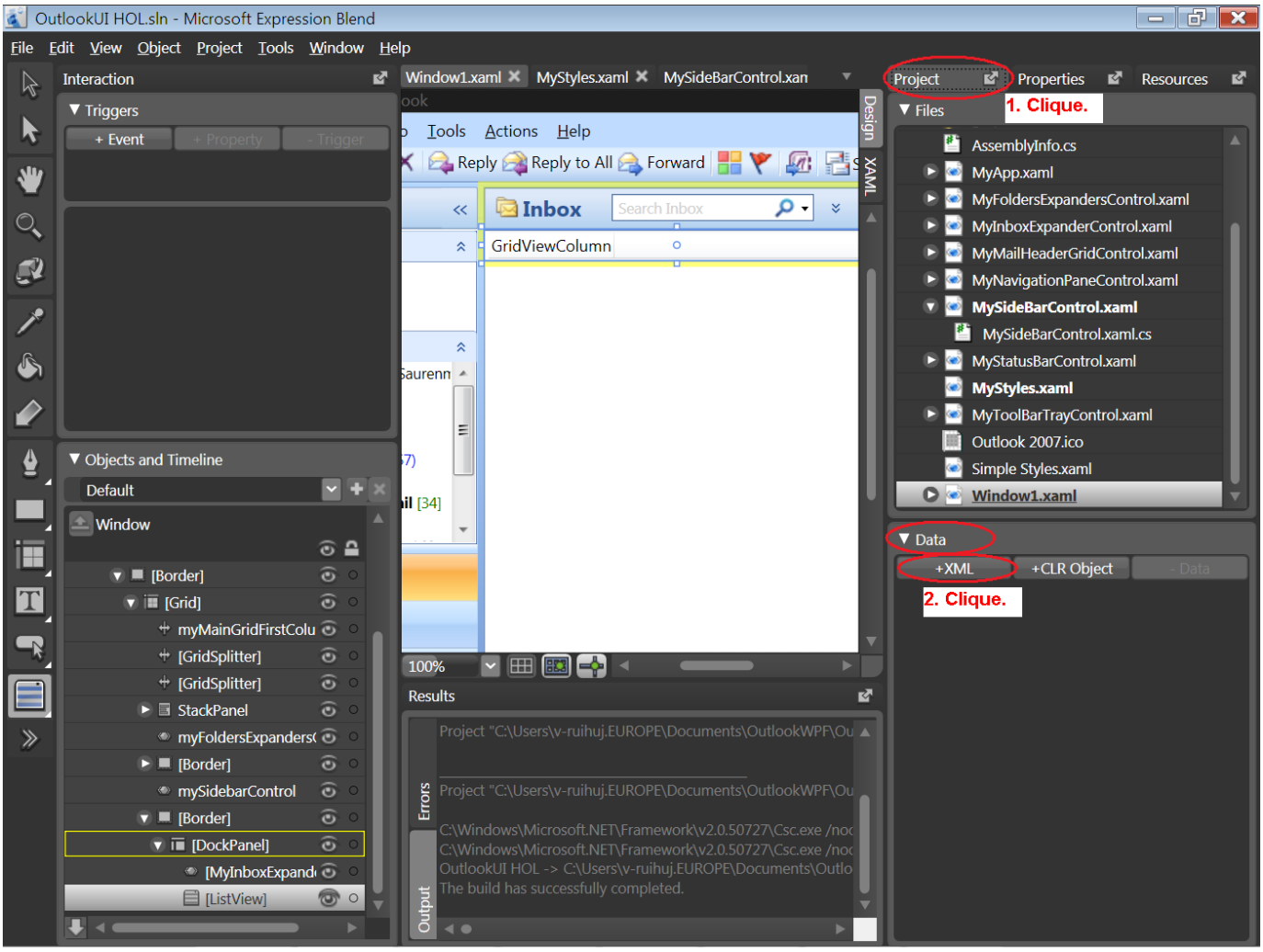


Figura 52 Abrindo a caixa de diálogo Add XML Data Source.

Na caixa de diálogo, defina o URL for XML data para **inbox.xml** (**inbox.xml** já foi configurado como **Resource** do projeto). Clique em **OK** para fechar a caixa de diálogo.

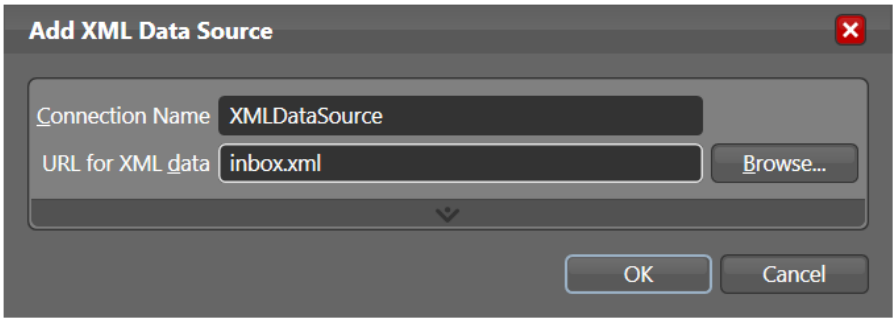


Figura 53 Definindo o URL for XML data.

Como pode ver, um novo data source chamado **inboxDS** foi adicionada no projeto. O elemento da raiz é **inbox**, dentro do elemento **inbox** é uma matriz de elementos **mail**, e cada elemento **mail** contém vários elementos filho que especificam as propriedades do mail relacionado (veja figura 54).

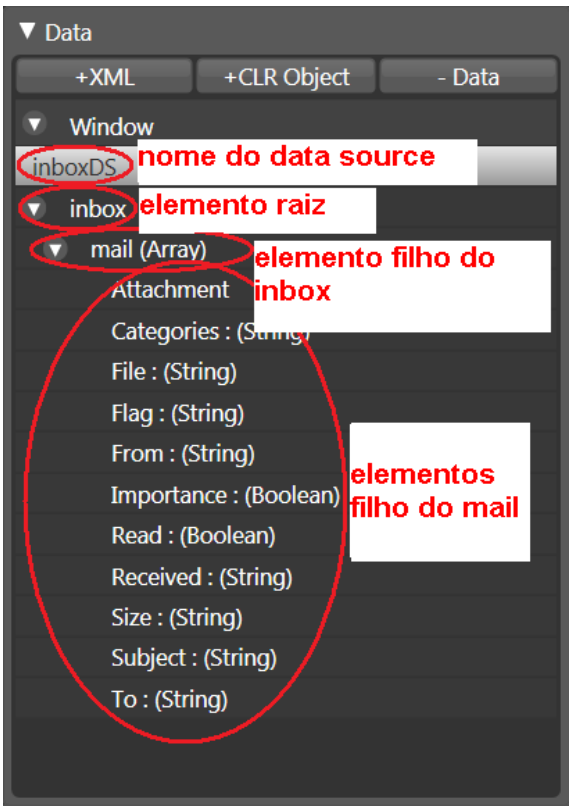


Figura 54 A estrutura de dados XML do inboxDS.

3. Faça um data bind com data context do **LayoutRoot** no **inboxDS**. **55**

Data Context

Data context é um conceito que permite elementos (como **[ListView]** no nosso Projeto) herdarem as informações de seus elementos pai (como **LayoutRoot** no nosso Projeto) sobre o data source que é usado para vinculação, bem como outras características de vinculação, como o path.

Selecione **LayoutRoot**, na categoria **Common Properties**, clique no pequeno quadrado ao lado da propriedade **DataContext** para abrir o menu contextual, selecione **Data Binding...**.
A caixa de diálogo **Create Data Binding** é aberta. Execute as etapas descritas na figura 55 para ligar a propriedade **LayoutRoot.DataContext** no **/inbox/mail do inboxDS**.

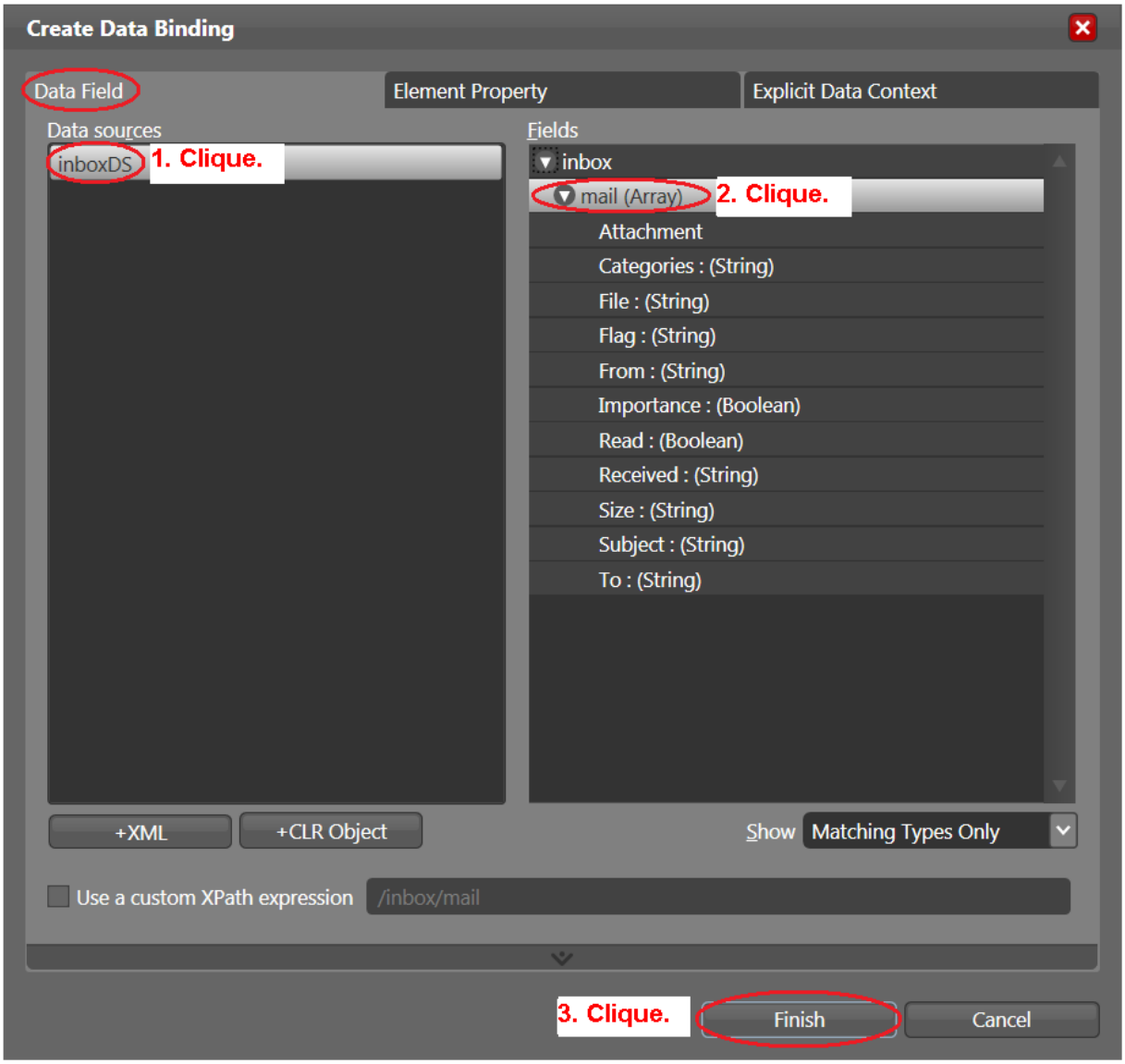


Figura 55 Vinculando dados LayoutRoot.DataContext no /inbox/mail do inboxDS.

Assim que tiver acoplado o **LayoutRoot.DataContext** a um XML data source, os elementos filho do **LayoutRoot** (como **[ListView]**) herdam automaticamente as informações do data source de seu controle pai. Isso pode ser confirmado se você for para o Properties Editor do **[ListView]** e consultar se ao lado da propriedade **DataContext**, o **(XmlDataCollection)** foi adicionado recentemente.

4. Faça um data bind da propriedade **[ListView].ItemsSource** para o contexto de dados especificado. **56**
Com **[ListView]** selecionado, clique no quadrado pequeno ao lado de propriedade **ItemsSource** na categoria **Common Properties**, selecione **Data Binding...** para abrir a caixa de diálogo **Create Data Binding**. Execute as etapas descritas na figura 56 para vincular **[ListView].ItemsSource** no elemento **mail** do contexto de dados especificado (você seleciona a opção “ Explicit Data Context ”, porque você está vinculando **ItemsSource** ao **DataContext** herdada do controle pai do **[ListView]**).

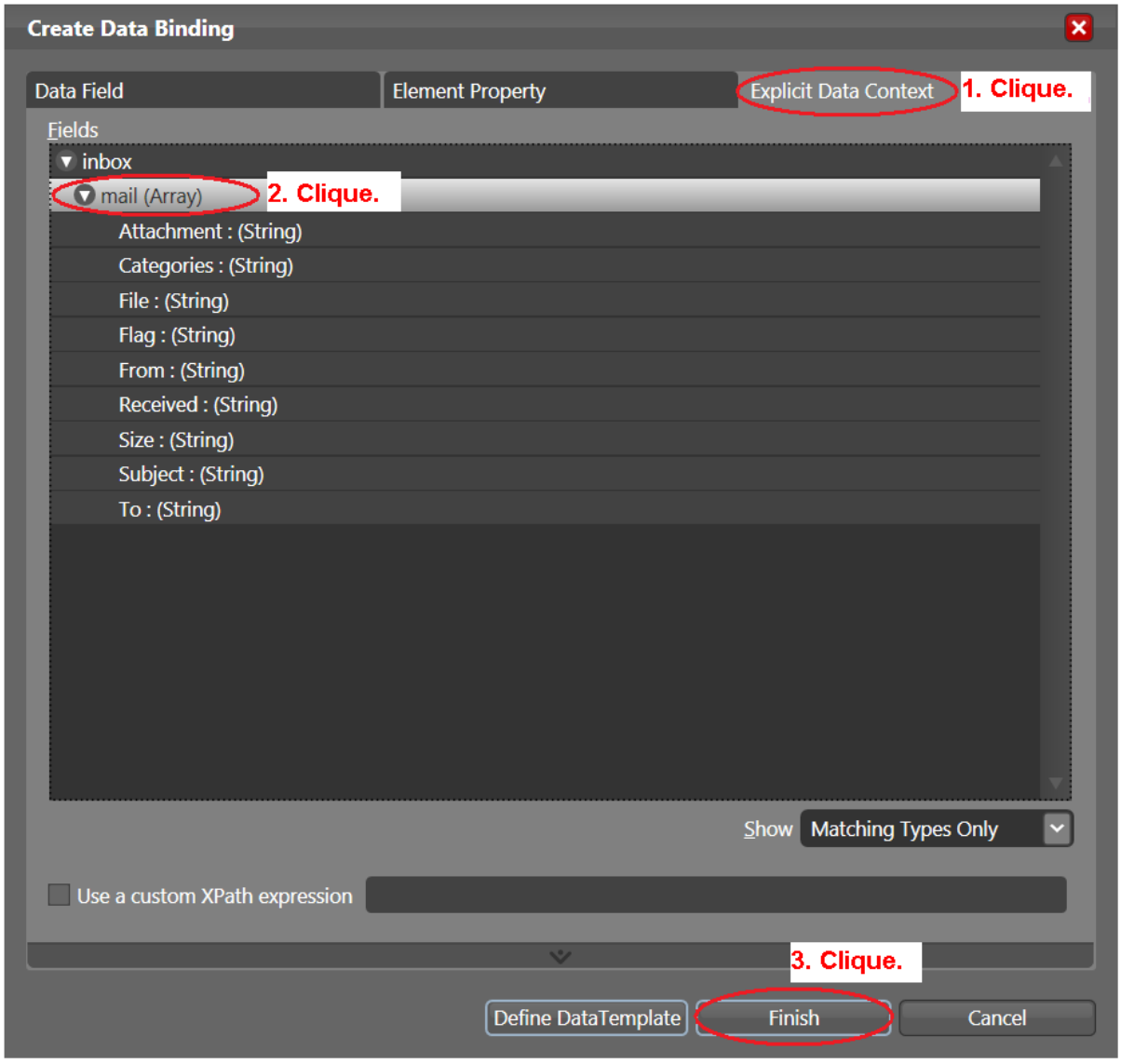


Figura 56 Vinculando dados [ListView].ItemsSource do contexto de dados especificado.

5. Crie colunas da lista para mostrar propriedades do mail como remetente, receptor e assunto:
Por padrão o **ListView.View** é um **GridView** que permite criar colunas e mostrar valores de uma determinada propriedade na coluna relacionada. Execute as seguintes etapas para criar três colunas **From**, **To** e **Subject**.
    1) Na categoria **Miscellaneous**, expanda o editor de propriedades **View (GridView)** e clique no botão para abrir a caixa de diálogo **GridViewColumn Collection Editor**. 57

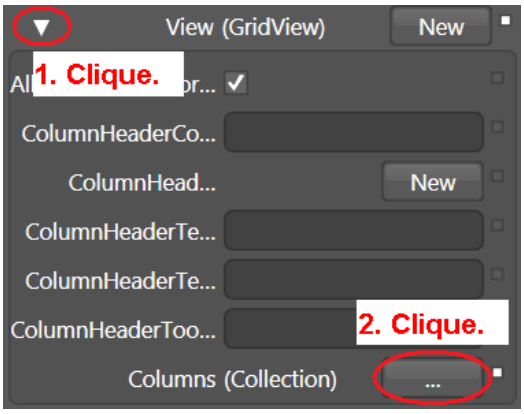


Figura 57 Abrindo o GridViewColumn Collection Editor.

2) Por padrão, já existe uma coluna chamada **GridViewColumn** no **GridView**. Defina o valor da propriedade de **Header** para **"From"**.

3) Adicione uma segunda coluna com **Header=To**. **58**

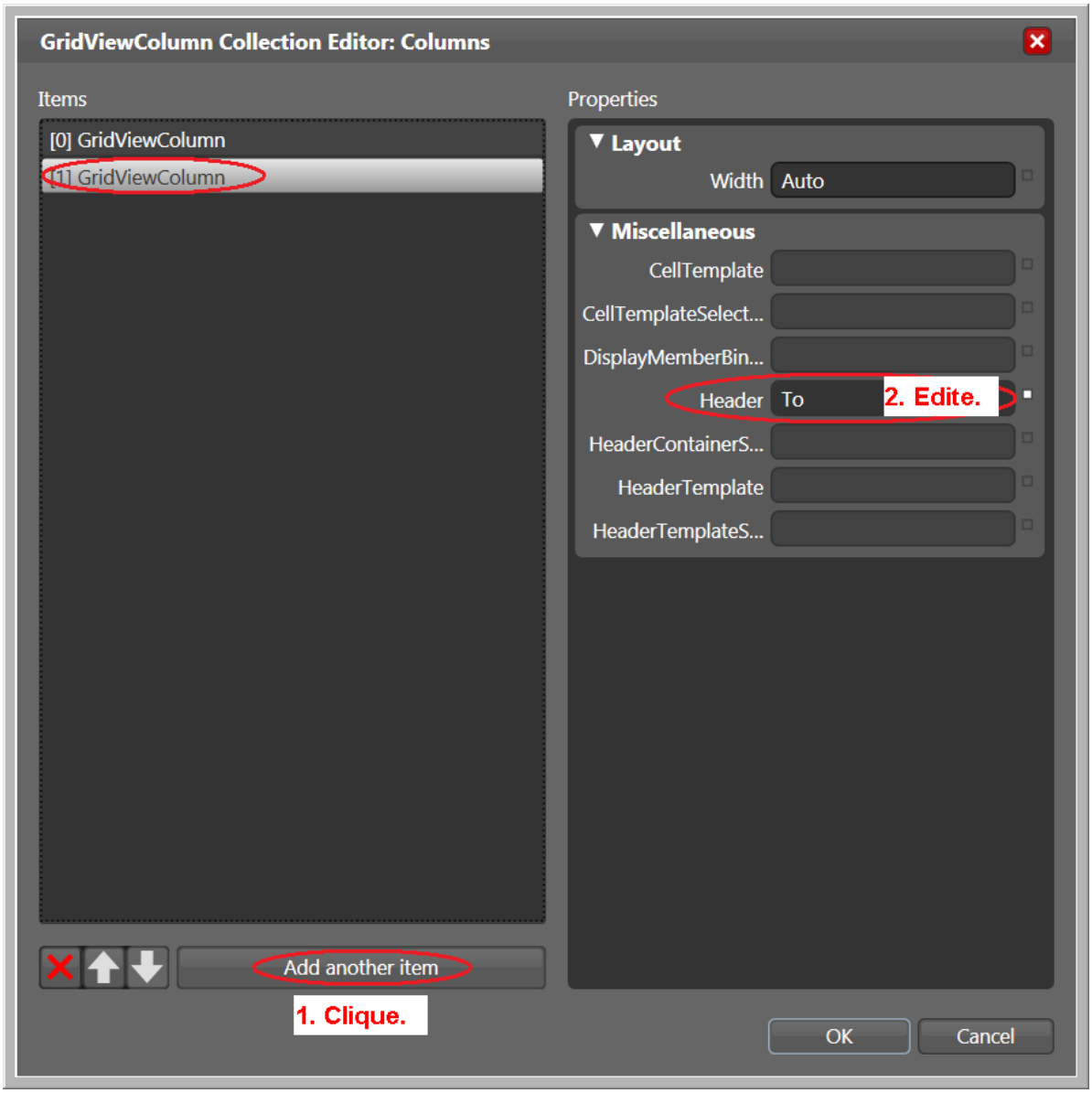


Figura 58 Adicionando uma segunda coluna com Header=To.

4) Adicione uma terceira coluna com **Header=Subject**.
5) Clique em **OK** para fechar a caixa de diálogo.
6) Faça um data bind da propriedade **DisplayMemberBinding** de cada coluna a uma propriedade mail.

Devido a um erro no Expression Blend não é possível fazermos um data binding para a propriedade **GridViewColumn.DisplayMemberBinding** no Expression Blend, que significa que nós temos que editar no arquivo XAML diretamente.

Em **Objects and Timeline**, clique com o botão direito do mouse no **[ListView]** para abrir o menu contextual, selecione **View XAML** para alternar para o modo XAML do **Window1.xaml**, pressione **F4** para obter uma exibição maior do documento. Como pode ver, o elemento **ListView** é realçado em azul em seu plano de fundo, e precisamos adicionar os seguintes textos (em vermelho) para fazer um data bind da propriedade **DisplayMemberBinding** de cada coluna dos elementos **From**, **To** e **Subject** do XML data source.

basic10.txt

```
<ListView DataContext="{Binding Path=DataContext, ElementName=LayoutRoot,
Mode=Default}" BorderBrush="{DynamicResource MyDarkBlueSolidBrush}"
BorderThickness="0,0.5,0,0" DockPanel.Dock="Top" ItemsSource="{Binding Mode=OneWay}"
IsSynchronizedWithCurrentItem="True">
  <ListView.View>
    <GridView>
        <GridViewColumn Header="From" DisplayMemberBinding="{Binding XPath=From}" />
        <GridViewColumn Header="To" DisplayMemberBinding="{Binding XPath=To}" />
        <GridViewColumn Header="Subject" DisplayMemberBinding="{Binding XPath=Subject}" />
    </GridView>
  </ListView.View>
</ListView>
```

Crie e execute o projeto (**F5**), certifique-se de que a lista exiba as propriedades **From**, **To** e **Subject** de cada mail.

Tarefa 11: Adicionar controles à terceira coluna do [Grid] no conteúdo Display Mail

A terceira coluna de **[Grid]** é usada para exibir os detalhes do mail atualmente selecionado.

1. Com **[Grid]** ativado, adicione um controle **DockPanel** no **[Grid]** e nomeie-o como **dp3**.
2. Agrupe o **DockPanel dp3** em um **Border**.
3. Configure as propriedades do **[Border]** da seguinte maneira:
   Brushes
   - **BorderBrush = MyDarkBlueSolidBrush**

   Appearance
   - **BorderThickness = 0.5, 0.5, 0.5, 0.5**

   Layout
   - **Width = Auto, Height = Auto**
   - Row = 0, **RowSpan = 2, Column = 2,** ColumnSpan = 1
   - **HorizontalAlignment = Stretch, VerticalAlignment = Stretch**
   - **Margin = 0, 0, 0, 0**
4. Configure as propriedades do **dp3** da seguinte maneira:
   Layout
   - **Width = Auto, Height = Auto**
   - HorizontalAlignment = Stretch, VerticalAlignment = Stretch
   - **LastChildFill = True**
   - Margin = 0, 0, 0, 0
5. Com **dp3** ativado, adicione um controle personalizado **MyMailHeaderGridControl** no **dp3** e configure suas propriedades da seguinte maneira:
   Layout
   - **Dock = Top**
   - HorizontalAlignment = Stretch, **VerticalAlignment = Top**

   Common Properties
   - DataContext = MS.Internal.Data.XmlDataCollection
     Como **[MyMailHeaderGridControl]** é também um controle filho de **LayoutRoot**, ele herda automaticamente as informações do data source de **LayoutRoot.DataContext**. No Properties Editor do **[MyMailHeaderGridControl]**, você pode ver o **DataContext (XmlDataCollection)**, isso porque nenhuma vinculação de dados explícita do **[MyMailHeaderGridControl].DataContext** se faz necessário.
     O **MyMailHeaderGridControl** tem quatro controles **TextBlocks** para exibir os elementos **Subject**, **From**, **Received** e **To** do mail correspondente. A ligação de dados das propriedades **TextBlock.Text** para **XPath=Subject**, **XPath=From**, **XPath=Received** e **XPath=To** já foram definidas no controle personalizado.

6. Com **dp3** ativado, adicione um controle **Frame** no **dp3** e configure suas propriedades da seguinte maneira:

Frame

Frame é um controle de conteúdo que fornece a capacidade de navegar e exibir o conteúdo. Conteúdo pode ser qualquer tipo de objeto .NET Framework 3.0 e arquivos HTML. No nosso projeto, os E-mails são arquivos HTML localizados sob o diretório **project folder\OutlookUI HOL\Initial Projects\Current Project\resources**.

Layout

- **Dock = Top**

Common Properties

- **NavigationUIVisiblity = Hidden**
- DataContext = MS.Internal.Data.XmlDataCollection (herdadas do **LayoutRoot.DataContext**)

7. Vincule os dados da propriedade **[Frame].Source** para **XPath=File** e converta cada elemento **File** em seu local de arquivo correspondente, o qual é o valor atual da propriedade **[Frame].Source**.
   Em **inbox.xml**, os arquivos para os mails são especificados como **<File>mail*.htm</File>** nos elementos **<mail>**. Como os locais de arquivo atuais para esses mails é **project folder\OutlookUI HOL\Initial Projects\Current Project\resources\mail*.htm**, é necessário primeiro converter **mail*.htm** para o path absoluto correspondente **project folder\OutlookUI HOL\Initial Projects\Current Project\resources\mail*.htm** antes de definir a propriedade **[Frame].Source**. Isso é feito usando um **Converter**.

Converter (conversores)

Para converter dados durante a vinculação, você deve criar uma classe que implementa a interface **IValueConverter**, que inclui os métodos **Convert** e **ConvertBack**. Os conversores podem alterar os dados de um tipo para outro, converter dados com base nas informações culturais ou modificar outros aspectos de apresentação.
Ao implementar a interface **IValueConverter**, é uma boa prática decorar a implementação com um atributo **ValueConversion** para indicar às ferramentas de desenvolvimento os tipos de dados que envolveram na conversão.

1) Adicione a seguinte classe **MyFrameSourceConverter** no namespace **OutlookUI_HOL** no **Window1.xaml.cs**:

basic11.txt

```
[ValueConversion(typeof(string), typeof(string))]
public class MyFrameSourceConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter,
                                        System.Globalization.CultureInfo culture)
    {
        string file = System.IO.Path.GetFullPath("..\\..\\.") + "\\resources\\" + (string)value;
        return file;
    }
    public object ConvertBack(object value, Type targetType, object parameter,
                                        System.Globalization.CultureInfo culture)
    {
        return null;
    }
}
```

2) Compile o projeto (**F5**) para fazer com que a classe acima seja compilada.
3) Adicione o seguinte texto (em vermelho) para o **Window.Resources** no **Window1.xaml** para registrar o conversor acima como um recurso:

basic11.txt

```
<Window … xmlns:local="clr-namespace:OutlookUI_HOL" … >
    <Window.Resources>
        <XmlDataProvider x:Key="inboxDS" d:IsDataSource="True" Source="inbox.xml"/>
        <local:MyFrameSourceConverter x:Key="MyFrameSourceConverter" />
    </Window.Resources>
```

4) Com **[Frame]** selecionado, clique no quadrado pequeno ao lado da propriedade **Source** para abrir o menu contextual, selecione **Custom Expression...** para abrir a caixa de diálogo, digite

basic11.txt

{Binding XPath=File, Converter={StaticResource MyFrameSourceConverter}}

na caixa de texto e pressione **Return**.

Crie e execute o aplicativo (**F5**). Selecione os itens da lista um após o outro, certifique-se que o **[MyMailHeaderGridControl]** e o controle **[Frame]** atualizem seus conteúdos conforme o esperado.

A seguir, têm um resumo do que nós fizemos nas tarefas 10 e 11, a fim de conseguir o **Master-Details View**:

1. Adicionar um XML data source no projeto.
2. Vincular dados do **LayoutRoot.DataContext** para o path **/inbox/mail** do XML data source do **inboxDS**. Todos os elementos de herança do **LayoutRoot**, então, herdam as informações do data source **LayoutRoot.DataContext** automaticamente. Suas propriedades **DataContext** estão todas implicitamente definidas no **MS.Internal.Data.XmlDataCollection**.
3. Vincular dados da propriedade **[ListView].ItemsSource** ao contexto explícito de dados.
4. Configurar a propriedade **[ListView].IsSynchronizedWithCurrentItem**, herdada do **Selector**, como **true** para garantir que **[MyMailHeaderGridControl]** e **[Frame]** sempre se referem ao item selecionado no **[ListView]**.

Propriedade Selector.IsSynchronizedWithCurrentItem

A propriedade **Selector.IsSynchronizedWithCurrentItem** obtém ou define um valor que indica se um **Selector** deve manter o **SelectedItem** sincronizado com o item atual na propriedade **Items**.
Você pode definir a propriedade **IsSynchronizedWithCurrentItem** para **true** para garantir que o item selecionado sempre corresponde à propriedade **CurrentItem** no **ItemCollection**. Por exemplo, suponha que há dois controles **ListView** com suas propriedades **ItemsSource** definidas para a mesma fonte. Defina **IsSynchronizedWithCurrentItem** como **true** em ambos os controles **ListView** para garantir que o item selecionado em cada **ListView** seja o mesmo.

5. Gravar declarações **Binding** para os elementos que exibem informações mail, como o conteúdo sender, receiver, subject e mail.
    1) Para o **GridViewColumn** do **GridView** no **[ListView]** temos declarações **Binding** como
    <GridViewColumn Header="To" DisplayMemberBinding="{Binding XPath=To}" />

    2) Para os controles **TextBlocks** no **[MyMailHeaderGridControl]** temos declarações **Binding** como
    <TextBlock Text="{Binding XPath=Subject}" />

    3) Para o controle **Frame** temos a declaração **Binding**
    <Frame Source="{Binding XPath=File}" />

Parabéns! Você fez um grande trabalho! Se estiver ansioso para mais recursos WPF, basta continuar com nosso Laboratório Prático. Nos seguintes cursos avançados, você aprenderá como personalizar um controle **Button** e um controle **ListView**.

Criando o Sidebar (avançado)

No curso básico foi-lhe solicitado para adicionar o controle personalizado **MySidebarControl** que consiste de dois botões. Pode parecer simples adicionar dois botões, entretanto, como você verá, há mais para ser feito para conseguir um sidebar estilo Outlook.

Neste curso avançado você aprenderá sobre

- personalizar o modelo de controle de um controle **Button**,
- vinculação de modelo,
- **ContentPresenter**,
- disparadores de propriedade de multi-condição, e
- gravar manipuladores de eventos para exibir ou ocultar o **MyNavigationPaneControl**.

Primeiro é necessário excluir o **mySidebarControl** antigo do **[Grid]**: Em **Objects and Timeline**, clique com o botão direito do mouse no **mySidebarControl** para abrir o menu contextual, escolha **Delete** para excluir o **mySidebarControl**.

Tarefa 1: Criar o Botão Sidebar

A figura 59 mostra a aparência do sidebar, sua estrutura hierárquica de controle aparece ilustrada na figura 60.

Figura 59 O sidebar.

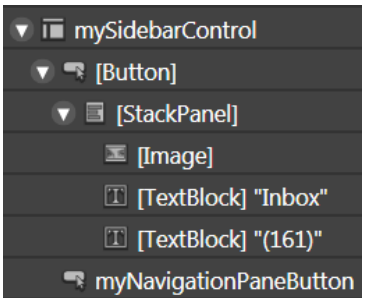


Figura 60 A estrutura hierárquica de controle do sidebar.

1. Com **[Grid]** ativado, adicione um controle **DockPanel** no **[Grid]**, configure suas propriedades da seguinte maneira:
   - **Name = mySidebarControl**

   Layout

   - **Width = Auto, Height = 27**
   - **HorizontalAlignment = Left, VerticalAlignment = Stretch**
   - **LastChildFill = True** (Marcado)

- **Margin: left = 0, right = 0, top = 32, bottom = 7**

2. Adicione um controle **Button** no **mySidebarControl**, configure suas propriedades da seguinte maneira:
   Brushes
   - **Background = MyBrightBlueSolidBrush2**
   - **BorderBrush = MyDarkBlueSolidBrush**

   Appearance
   - **BorderThickness = 0.5, 0.5, 0.5, 0.5**
3. Adicione um controle **StackPanel** no **[Button]**, configure suas propriedades da seguinte maneira:
   Layout
   - **Width = Auto, Height = Auto**
   - **Orientation = Horizontal**
4. Adicione um controle **Image** no **[StackPanel]**, configure suas propriedades da seguinte maneira:
   Layout
   - **Width = 16, Height = 16**
   - **HorizontalAlignment = Center, VerticalAlignment = Center**
   - **Margin: left = 4, right = 2,** top = 0, bottom = 0

   Common Properties
   - **Source = graphics\inbox.gif**
   - **Stretch = Fill**
5. Adicione um controle **TextBlock** no **[StackPanel]**, configure suas propriedades da seguinte maneira:
   Brushes
   - **Foreground = MyDarkBlueSolidBrush**

   Layout
   - **HorizontalAlignment = Center, VerticalAlignment = Center**
   - **Margin: left = 2, right = 2,** top = 0, bottom = 0

   Common Properties
   - **Text = Inbox**

   Text
   - **FontWeight = Bold**
   - **TextWrapping = NoWrap**
6. Adicione um controle **TextBlock** no **[StackPanel]**, configure suas propriedades da seguinte maneira:
   Brushes
   - **Foreground = #FF0000FF** (um solid color brush azul)

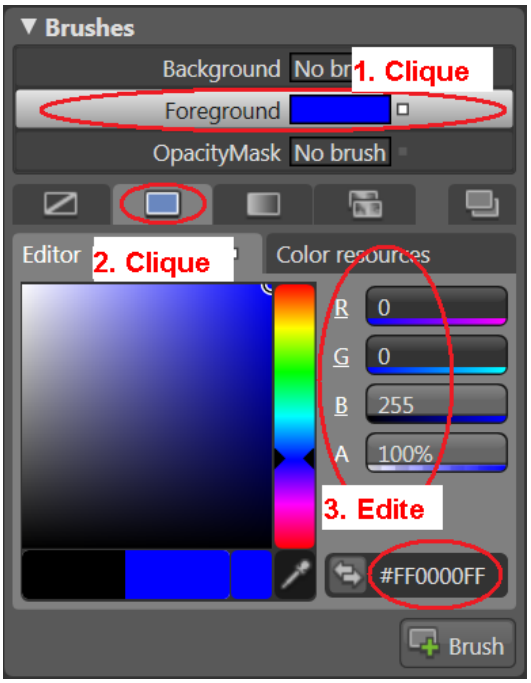


Figura 61 Definindo a propriedade Foreground do controle TextBlock.

Layout

- **HorizontalAlignment = Center, VerticalAlignment = Center**
- **Margin: left = 2, right = 4,** top = 0, bottom = 0

Common Properties

- **Text = (161)**

Text

- **TextWrapping = NoWrap**

7. Adicione um controle **Button** no **mySidebarControl**, configure suas propriedades da seguinte maneira:
   - **Name = myNavigationPaneButton**

   Brushes
   - **Background = MyBrightBlueSolidBrush2**
   - **BorderBrush = MyDarkBlueSolidBrush**
   - **Foreground = MyDarkBlueSolidBrush**

   Appearance
   - **BorderThickness = 0.5, 0.5, 0.5, 0.5**

   Layout
   - **Dock = Right**

   Common Properties
   - **Content = Navigation Pane**

   Text
   - **FontSize = 15**
   - **FontWeight = Bold**
8. Gire o controle **mySidebarControl** vertical.
   Com **mySidebarControl** selecionado, expanda a categoria **Transform**, sob a subcategoria **LayoutTransform**, defina as propriedades a seguir:
   - **Rotate : Angle = -90**
9. Gire o controle **[Image]**, criado na etapa 4, 90 graus no sentido horário:
   Com o controle **[Image]** selecionado, expanda a categoria **Transform**, sob a subcategoria **LayoutTransform**, defina as propriedades a seguir:
   - **Rotate : Angle = 90**

Crie e execute o aplicativo (**F5**). Mova o mouse sobre os botões no sidebar e clique neles. Como pode ver, os dois botões ter uma aparência diferente dos botões do Outlook, vamos começar a configurar nosso próprio modelo de controle para os dois botões.

Tarefa 2: Editar um Modelo de Controle para os Botões Sidebar

1. Abra o editor do modelo de controle para o **[Button]**. 62 63
   Clique com o botão direito do mouse no **[Button]** para abrir o menu contextual, selecione **Edit Control Parts (Template) > Edit a Copy…**. Na caixa de diálogo, nomeie o estilo **MySidebarButtonStyle**, e escolha **MyStyles.xaml** como o resource dictionary.

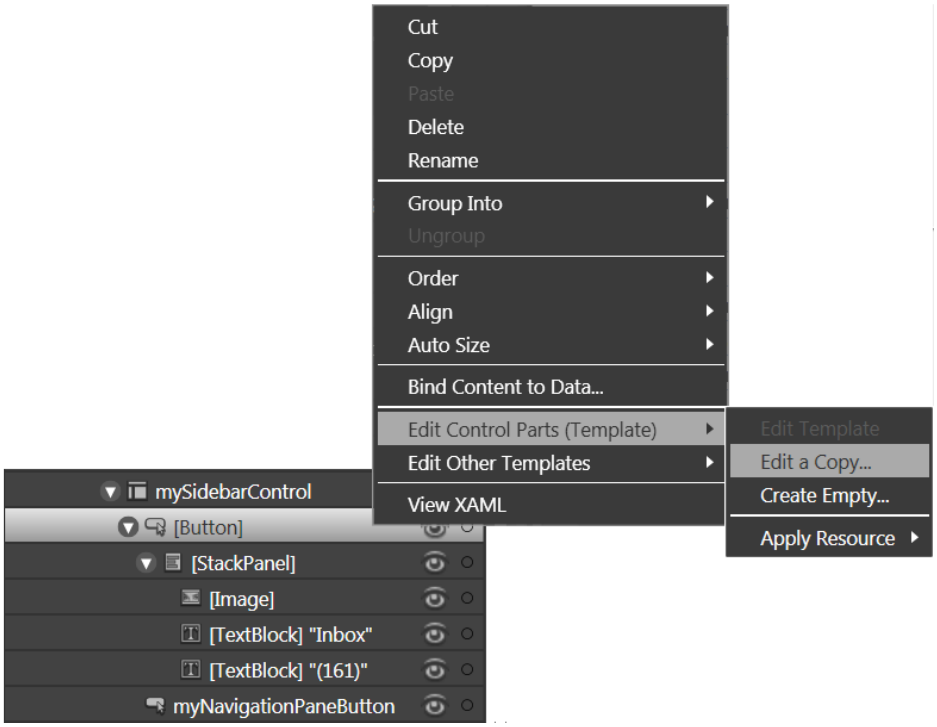


Figura 62 Abrindo o editor do modelo de controle para o controle [Button].

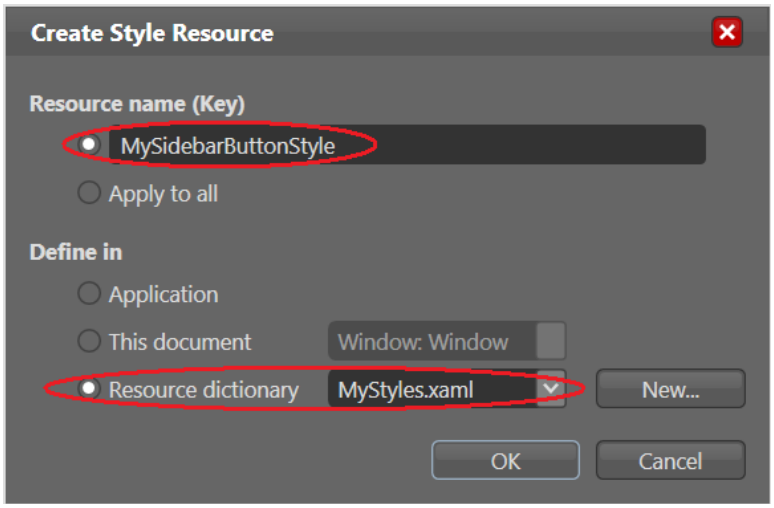


Figura 63 Definindo o nome e o resource dictionary para o estilo MySidebarButtonStyle.

O modelo de controle padrão para um botão tem a estrutura de controle conforme mostrado na figura 64:

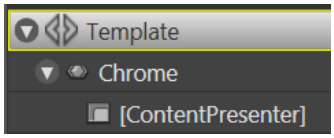


Figura 64 A estrutura de controle do estilo padrão de botão.

Para nosso projeto, queremos que a estrutura de controle aparente como o seguinte:

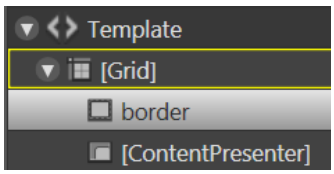


Figura 65 A estrutura de controle do nosso estilo de botão personalizado.

2. Em **Objects and Timeline**, clique com o botão direito do mouse no **Chrome** para abrir o menu contextual, escolha **Delete** para excluir **Chrome**.
3. Adicione um controle **Grid** no **Template**.
4. Adicione um controle **Border** que você pode localizar no **Asset Library** no controle **[Grid]**, configure suas propriedades como segue:
   Brushes

   - **Background = {TemplateBinding Background}** **66**
     Clique no quadrado pequeno ao lado da propriedade **Background** para abrir o menu contextual, escolha o item de submenu **Template Binding > Background**.

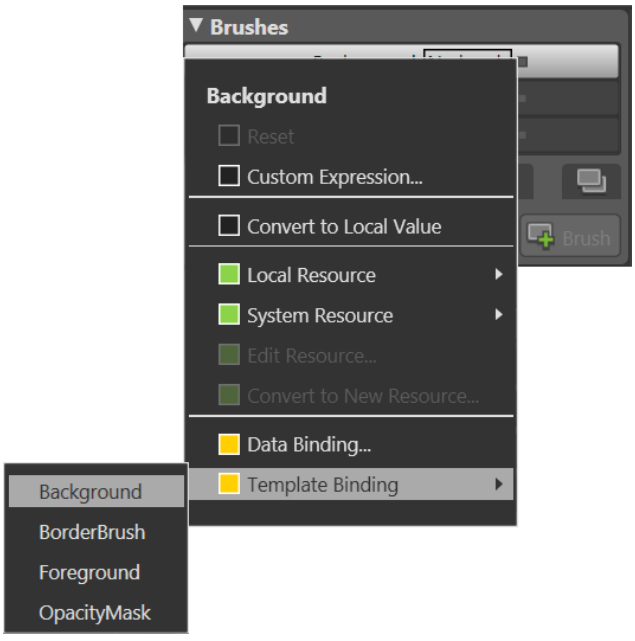


Figura 66 Ligando o template para a propriedade Background.

O pequeno quadrado ao lado de propriedade **Background** quando ativado em laranja, indica que a propriedade **Background** é um template-ligado.

Template Binding (Ligação de Template)

Template binding é usado para vincular as propriedades no template às propriedades do controle ao qual o template é aplicado. No nosso caso, template binding do **Border.Background** para o **Button.Background** tem o efeito que quando definimos a propriedade **Background** de um botão cujo estilo é definido como **MySidebarButtonStyle**, estamos realmente nos referindo à propriedade **Background** do controle **Border** no modelo de controle.

- **BorderBrush = {TemplateBinding BorderBrush}**

Appearance

- **BorderThickness = {TemplateBinding BorderThickness}**

Layout

- **Margin = 0, 0, 0, 0**

5. Adicione um controle **ContentPresenter** que você pode localizar no **Asset Library** no controle **[Grid]**, configure suas propriedades como segue:

Layout

- **HorizontalAlignment = Center, VerticalAlignment = Center**
- **Margin = 0, 0, 0, 0**

Content Presenter

Um content presenter é um espaço reservado no modelo de controle para exibir o conteúdo do controle ao qual o modelo é aplicado. Isso pode ser o valor de uma propriedade **Content** (em um **Button**, por exemplo), ou uma propriedade **Text** (em um **TextBox**).

6. Adicione e personalize o disparador de propriedade **IsPressed = True**. 67 68

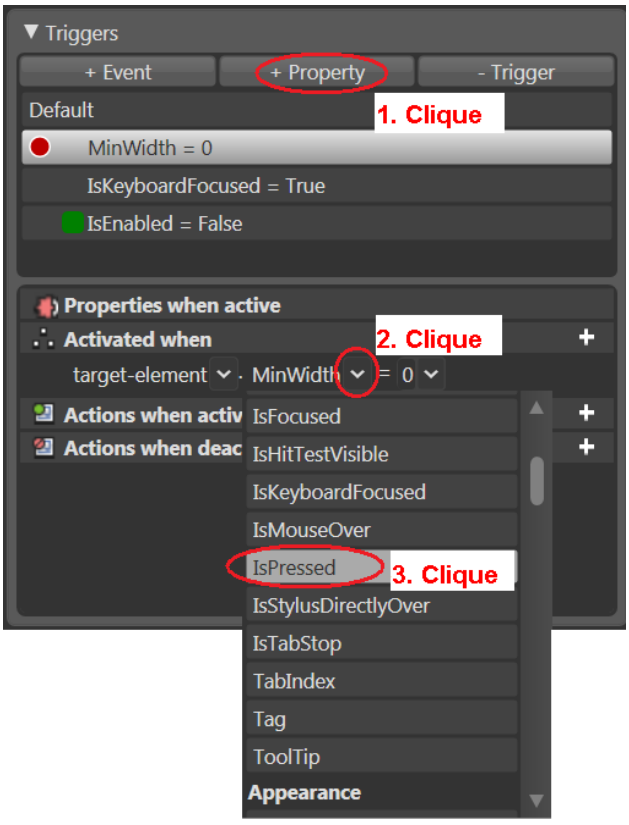


Figura 67 Adicionando o disparador de propriedade IsPressed = True.

Logo que um disparador de propriedade é adicionado, sua gravação é ativada, que registra todas as alterações de propriedades caso o disparador de propriedade esteja ativado.
Defina a propriedade **Background** do controle **border** para **MyDarkOrangeSolidBrush**, e pare a gravação do disparador (veja figura 68).

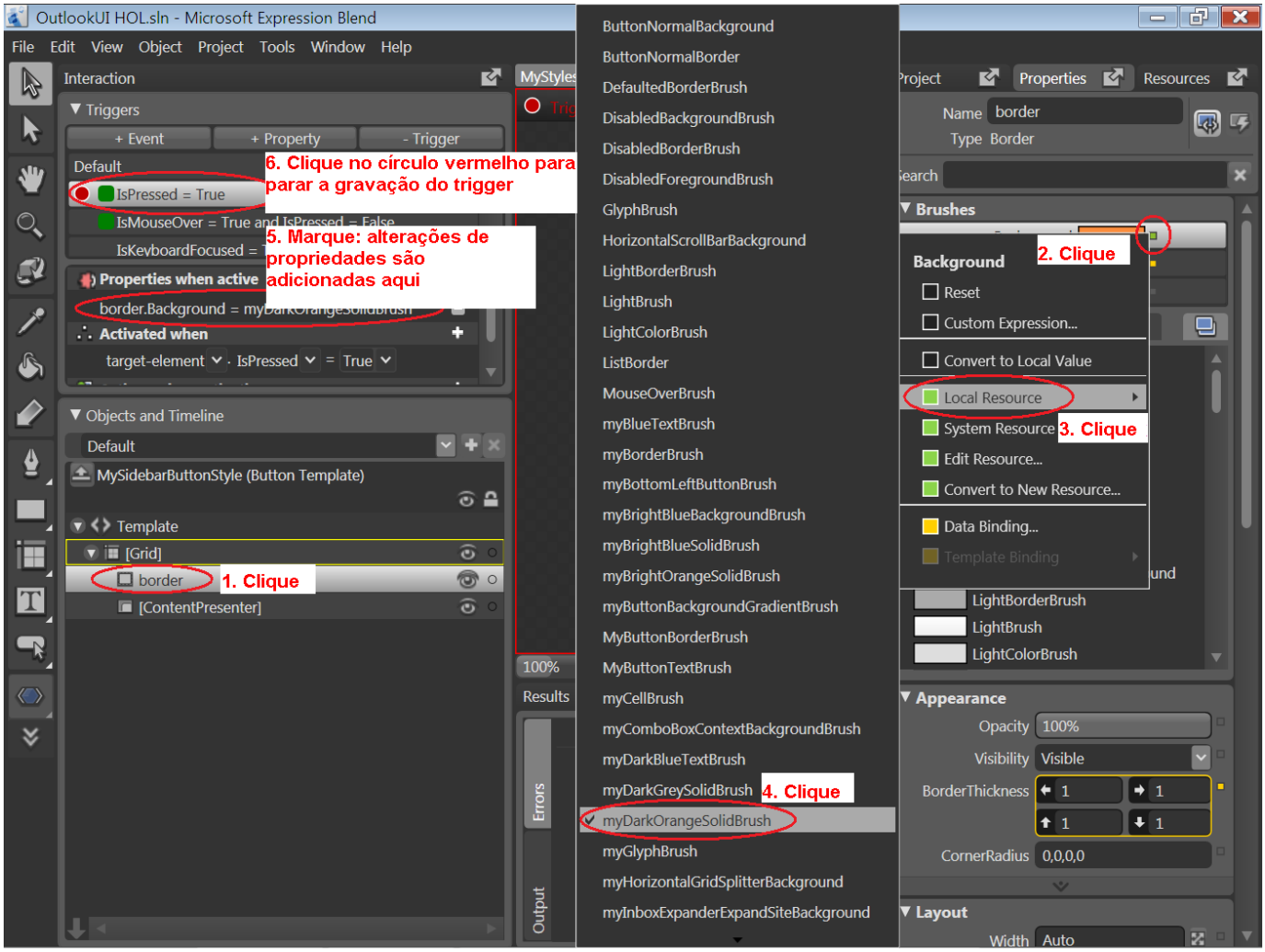


Figura 68 Personalizando o disparador de propriedade IsPressed = True.

7. Adicione e personalize o disparador de propriedade **IsMouseOver = True e IsPressed = False**.
   Primeiro adicione um disparador de propriedade **IsMouseOver = True**, conforme descrito para **IsPressed = True** (veja figura 67), em seguida, clique no botão no

   , então altere **MinWidth** para **IsPressed** e altere **True** para **False**. Dessa maneira você obtém uma expressão Booleana como **e1 e e2**, no nosso caso, a expressão significa que o disparador de propriedade está ativado, quando o usuário move o mouse sobre o botão e não clica nele.
   Com a gravação do disparador ligada, defina a propriedade **Background** do controle **border** para **MyBrightOrangeSolidBrush** e pare a gravação do disparador.

8. Em **Objects and Timeline**, clique para sair do editor de template de controle para **MySidebarButtonStyle**.

9. Aplique o **MySidebarButtonStyle** no **myNavigationPaneButton**. **69**
   Com **myNavigationPaneButton** selecionado, defina a propriedade **Style** da categoria **Miscellaneous** como **MySidebarButtonStyle**.

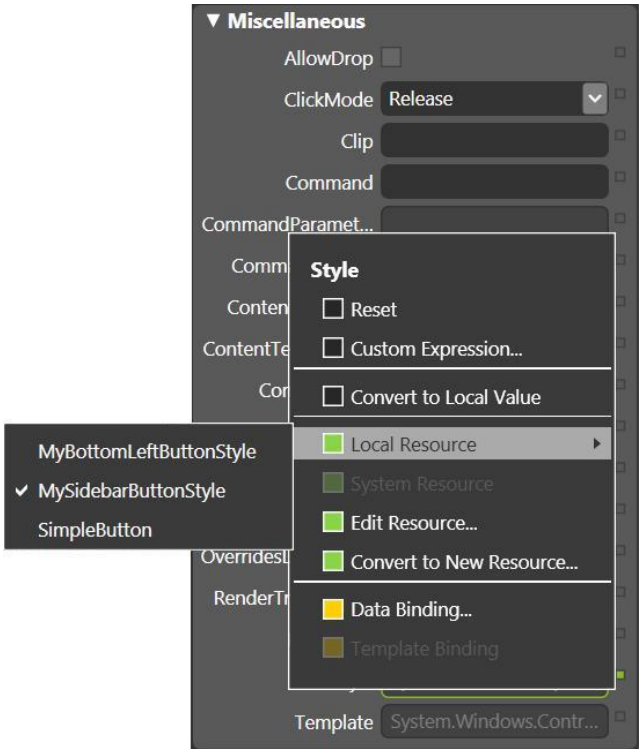


Figura 69 Definindo a propriedade Style do myNavigationPaneButton para MySidebarButtonStyle.

Crie e execute o aplicativo (**F5**). Mova o mouse sobre os botões e clique nos botões, certifique-se de que os botões têm cores de plano de fundo diferente em estados diferentes.

10. Na categoria **Appearance**, defina a propriedade **Visibility** do **mySidebarControl** como **Hidden**.

Tarefa 3: Escrever Manipuladores de Eventos para Mostrar/Ocultar o Painel de Navegação

1. Adicione o controle personalizado **MyNavigationPaneControl** no **[Grid]**, configure suas propriedades da seguinte maneira:
   - **Name = myNavigationPaneControl**

Appearance

   - **Visibility = Hidden**

Layout

   - **Width = Auto,** Height = Auto
   - Row = 0, **RowSpan = 2,** Column = 0, **ColumnSpan = 3**
   - **Margin: left = 27,** right = 0, **top = 32,** bottom = 0
2. Escreva um manipulador de eventos para o evento **myNavigationPaneButton.Click** e adicione uma lógica de programação adicional no **Window1.xaml.cs**:

sidebar.txt

```
private MouseButtonEventHandler mbEventHandler = null;
private DependencyPropertyChangedEventHandler visibilityEventHandler = null;

private void myNavigationPaneButton_Click(object sender, RoutedEventArgs e)
{
  if (mbEventHandler == null) {
        mbEventHandler = new MouseButtonEventHandler(LayoutRoot_PreviewMouseLeftButtonUp);
  }

  if (visibilityEventHandler == null) {
        visibilityEventHandler = new
        DependencyPropertyChangedEventHandler(myNavigationPaneControl_IsVisibleChanged);
```

```
    }
    if (myNavigationPaneControl.Visibility == Visibility.Hidden) {
        myNavigationPaneButton.SetValue(Button.BackgroundProperty,
                (Brush)MyApp.Current.Resources["MyOrangeSolidBrush"]);
        myNavigationPaneControl.Visibility = Visibility.Visible;
        LayoutRoot.PreviewMouseLeftButtonUp += mbEventHandler;
        myNavigationPaneControl.IsVisibleChanged += visibilityEventHandler;
    } else {
        hideNavigationPane();
    }
}

private void myNavigationPaneControl_IsVisibleChanged(object sender,
DependencyPropertyChangedEventArgs e)
{
    /* if myNavigationPaneControl becomes hidden, then
        * the look of the application UI is to be reset and
        * the two event handlers are to be deregistered, which is done in the method
        * hideNavigationPane().
        */
    if (myNavigationPaneControl.Visibility == Visibility.Hidden) {
        hideNavigationPane();
    }
}

private void LayoutRoot_PreviewMouseLeftButtonUp(object sender,
System.Windows.Input.MouseButtonEventArgs e)
{
    /* if the user clicks somewhere on LayoutRoot except on myNavigationPaneButton
        * and except on myNavigationPaneControl, and
        * myNavigationPaneControl is visible, then
        * myNavigationPaneControl is to be hidden.
        */
    if (e.OriginalSource != myNavigationPaneButton && e.Source != myNavigationPaneControl) {
        if (myNavigationPaneControl.Visibility == Visibility.Visible) {
            hideNavigationPane();
        }
    }
}

private void hideNavigationPane()
{
    myNavigationPaneButton.SetValue(Button.BackgroundProperty,
            (Brush)MyApp.Current.Resources["MyBrightBlueSolidBrush2"]);
    myNavigationPaneControl.Visibility = Visibility.Hidden;
    LayoutRoot.PreviewMouseLeftButtonUp -= mbEventHandler;
    myNavigationPaneControl.IsVisibleChanged -= visibilityEventHandler;
}
```

Personalizando o Controle [ListView] (avançado)

No curso básico, nós adicionamos itens de lista para o controle **[ListView]** sem considerar o estilo do **[ListView]**. O estilo padrão de um controle **ListView** é semelhante ao seguinte:

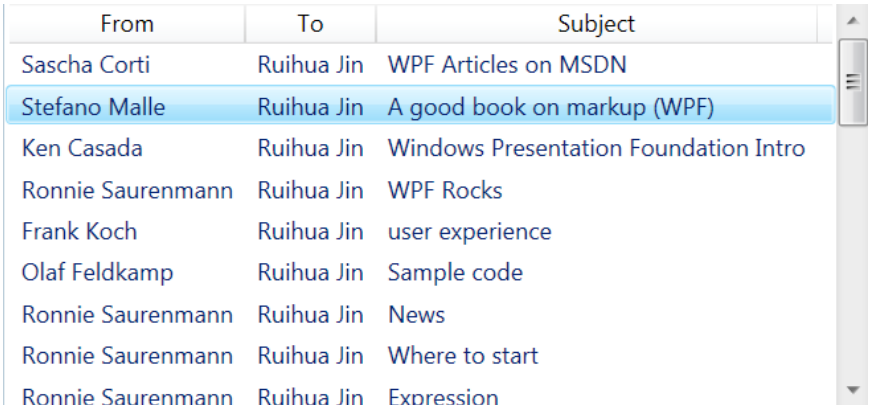


Figura 70 O controle [ListView] com o estilo padrão.

Nosso objetivo é atingir o estilo MS Outlook 2007:

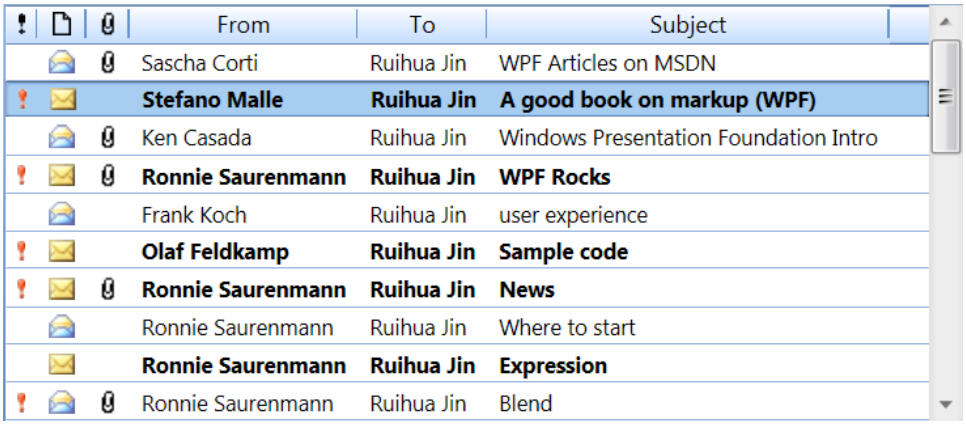


Figura 71 O ListView estilo Outlook.

Neste curso avançado você aprenderá sobre

- personalizar o estilo de cabeçalhos de coluna
- personalizar o estilo de itens de lista
- adicionar imagens nas colunas para representar dados
- classificar a lista

Entre os demais, dois novos conceitos importantes, envolvidos, são

- data triggers (disparadores de dados),
- data templating (modeladores de dados)

Tarefa 1: Editar o estilo dos cabeçalhos de colunas

1. Entre no **Template Editor** no **[ListView]**.
   Em **Objects and Timeline**, clique com o botão direito do mouse no **[ListView]** para abrir o menu contextual, escolha **Edit Control Parts (Template) > Edit a Copy…**. Na caixa de diálogo, nomeie o estilo **MyListViewStyle**, e escolha **MyStyles.xaml** como resource dictionary.
   A estrutura de controle padrão de um controle **ListView** é semelhante ao seguinte:

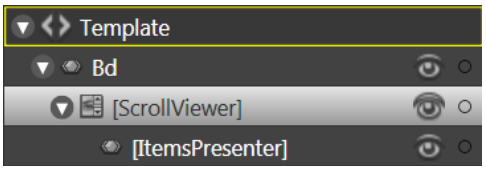


Figura 72 A estrutura de controle padrão do controle [ListView].

2. Edite **MyListViewScrollViewerStyle**.
   1) Entre no **Template Editor** no **[ScrollViewer]**.
      Em **Objects and Timeline**, clique com o botão direito do mouse no **[ScrollViewer]** para abrir o menu contextual, escolha **Edit Control Parts (Template) > Edit a Copy…**. Na caixa de diálogo, nomeie o estilo **MyListViewScrollViewerStyle** e escolha **MyStyles.xaml** como resource dictionary.
      A estrutura de controle padrão de um controle **ScrollViewer** é semelhante ao seguinte:

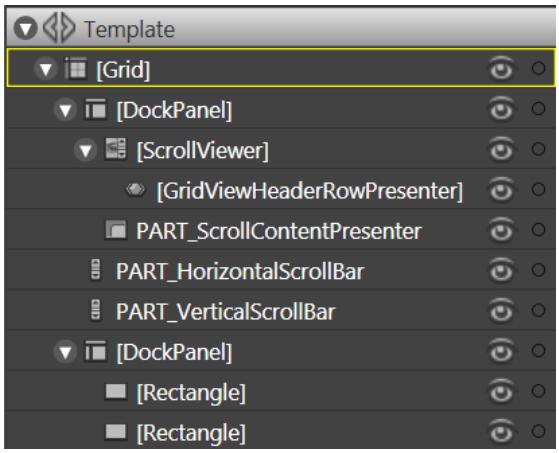


Figura 73 A estrutura de Controle do controle [ScrollViewer].

   2) Configure as propriedades do **[GridViewHeaderRowPresenter]** da seguinte maneira:
      Layout
      - **Margin = 0, 0, 0, 0**

3. Edite **MyGridViewColumnHeaderStyle**.
   1) Entre no **Style Editor** no **ColumnHeaderContainerStyle**. **74**

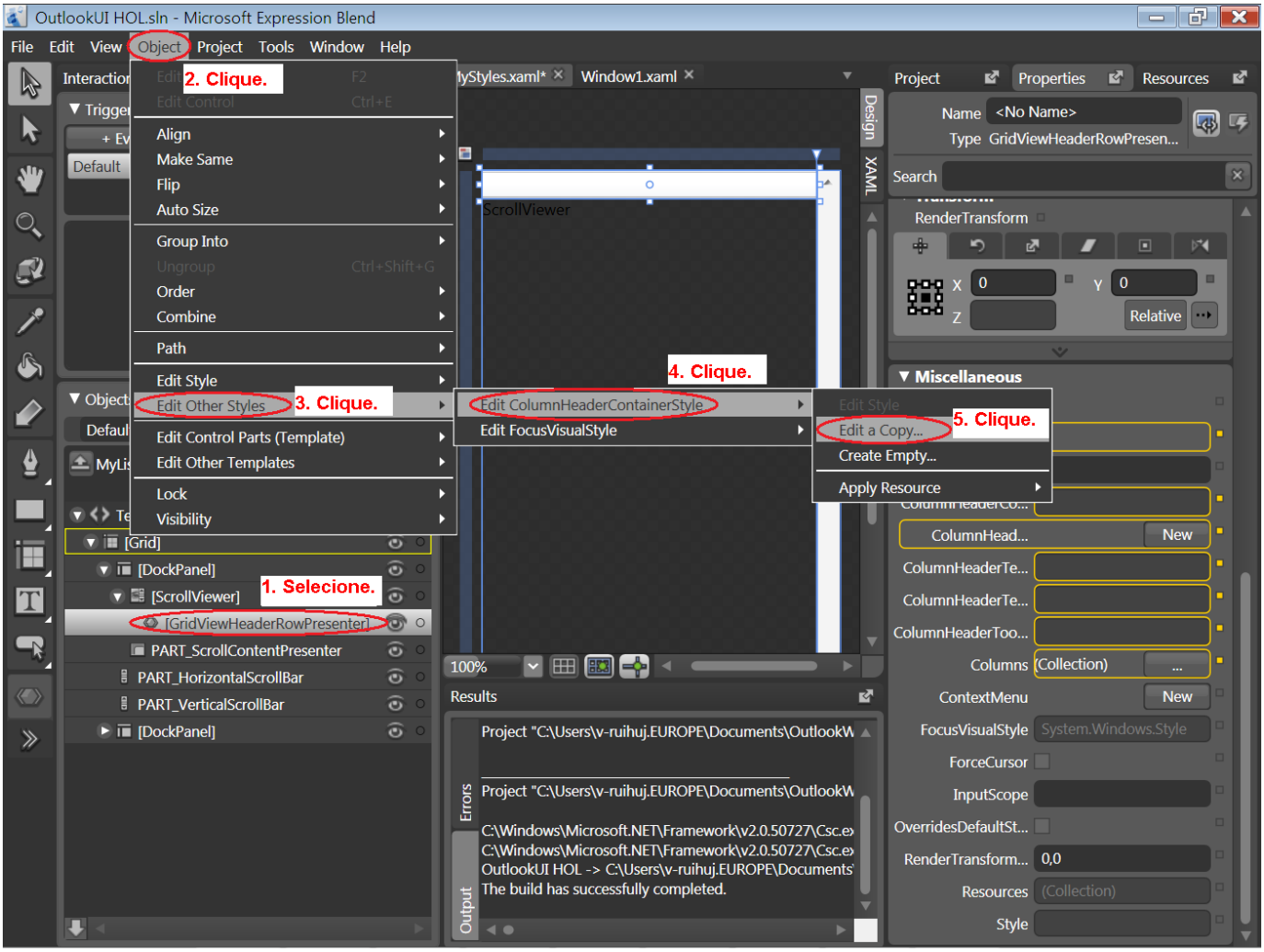


Figura 74 Inserindo o Style Editor no ColumnHeaderContainerStyle.

Na caixa de diálogo, nomeie o estilo **MyGridViewColumnHeaderStyle** e escolha **MyStyles.xaml** como resource dictionary.

2) Configure as propriedades do **MyGridViewColumnHeaderStyle** da seguinte maneira:

Brushes

- **Background = MyGridViewColumnHeaderBlueGradientBrush**
- **BorderBrush = MyDarkBlueSolidBrush**

Appearance

- **BorderThickness:** left = 0, right = 0, **top = 0.5, bottom = 0.5**

4. Edite **MyGridViewColumnHeaderControlTemplate**.

1) Entre no **Template Editor** no **MyGridViewColumnHeaderStyle**.
Em **Objects and Timeline**, clique com o botão direito do mouse em **Style** para abrir o menu contextual, escolha **Edit Control Parts (Template) > Edit a Copy…**. Na caixa de diálogo, nomeie o modelo de controle **MyGridViewColumnHeaderControlTemplate** e escolha **MyStyles.xaml** como resource dictionary. A estrutura de controle padrão do **MyGridViewColumnHeaderControlTemplate** é semelhante ao seguinte:

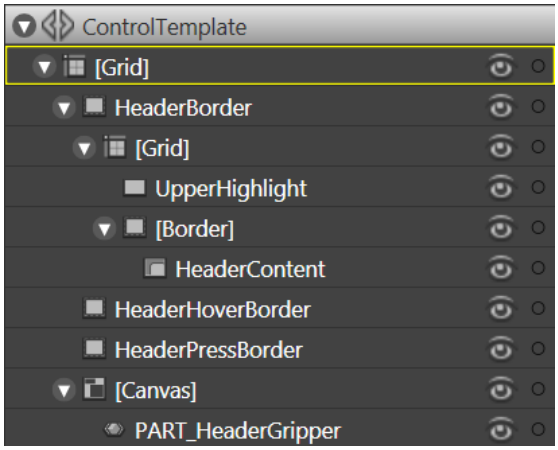


Figura 75 A estrutura de controle padrão do MyGridViewColumnHeaderControlTemplate.

2) Exclua **UpperHighlight**, **HeaderHoverBorder** e **HeaderPressBorder**.
3) Configure as propriedades do **HeaderBorder** da seguinte maneira:
Brushes
   - **Background = MyGridViewColumnHeaderBlueGradientBrush**

Appearance
   - **BorderThickness = 0, 0, 0, 0**

4) Configure as propriedades do **[Border]**, que é o controle pai do **HeaderContent**, da seguinte maneira:
Brushes
   - **BorderBrush = MyDarkBlueSolidBrush**

Appearance
   - **BorderThickness:** left = 0, right = 0, **top = 0.5, bottom = 0.5**

5) Configure as propriedades do **PART_HeaderGripper**, que é usado para denotar o final de um cabeçalho de coluna, da seguinte maneira:
Brushes
   - **Background = MyBlueSolidBrush1**

Layout
   - **Width = 1, Height = 16**
   - **Left = {Binding Path=ActualWidth, ElementName=Grid, Mode=Default}** **76**

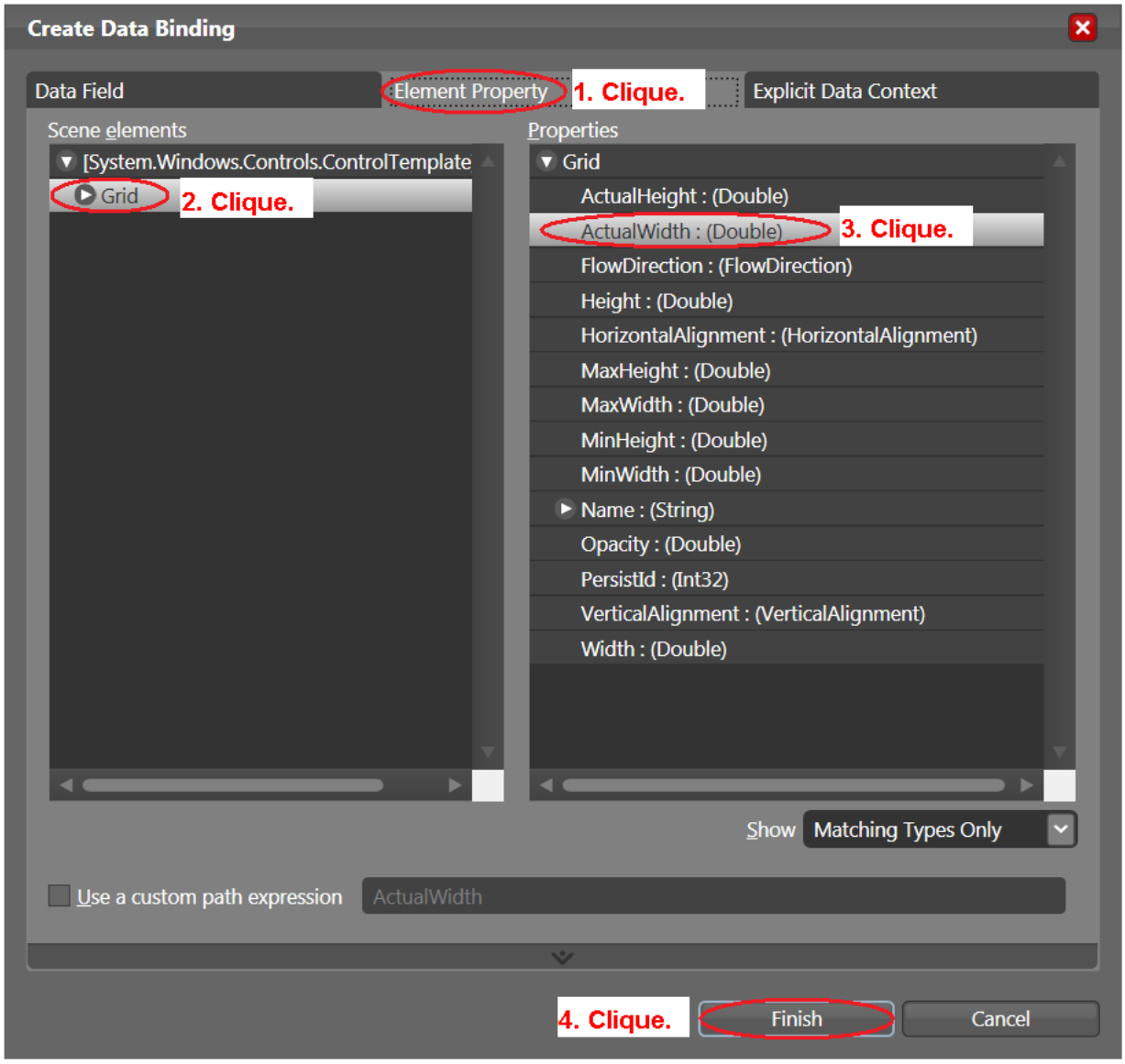


Figura 76 Vinculando dados PART_HeaderGripper.Left para Grid.ActualWidth.

- **Margin: left = -1, right = 0, top = 2, bottom = 2**

6) Exclua as alterações de propriedade do disparador de propriedade **IsMouseOver = True**. 77

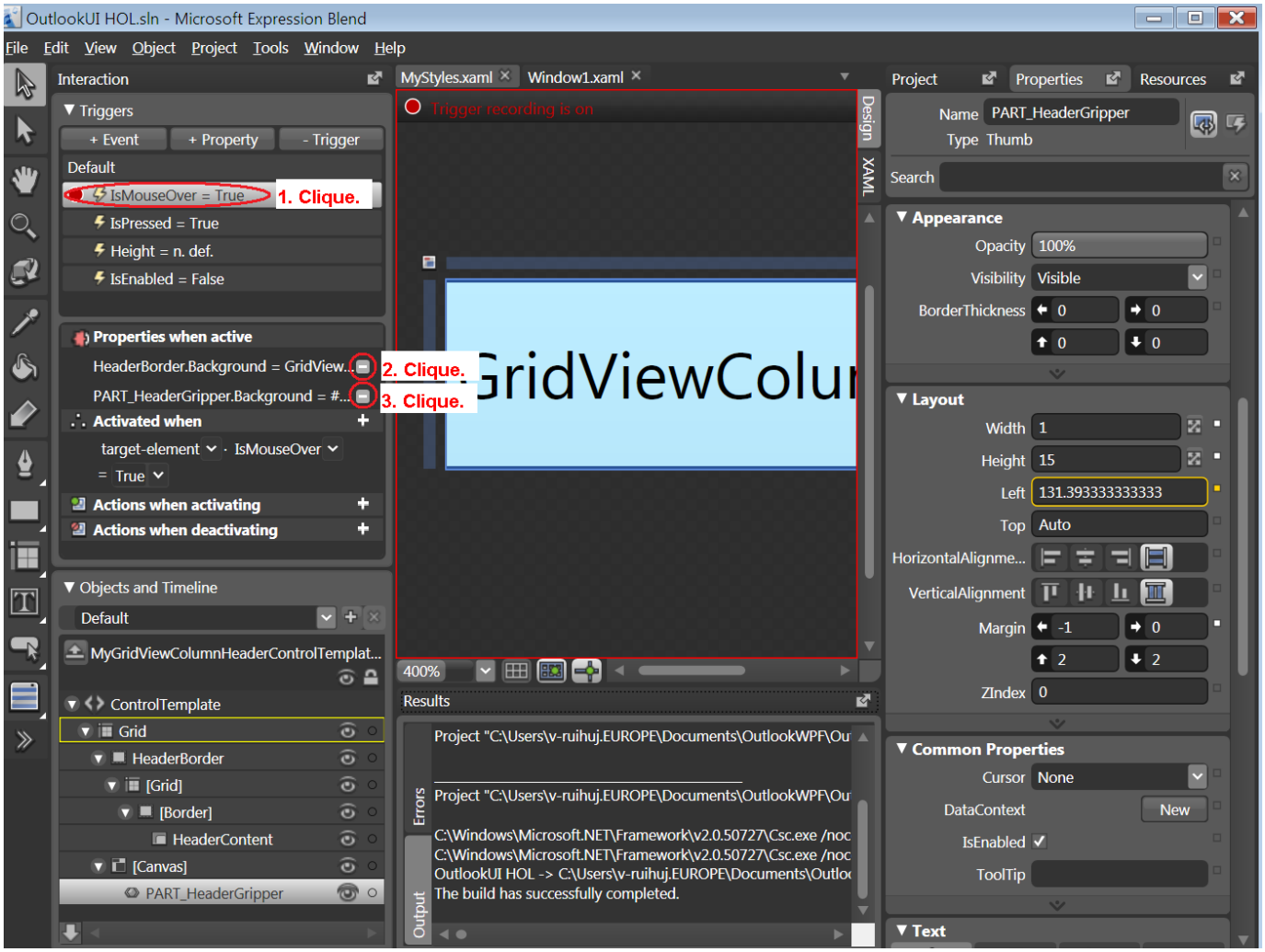


Figura 77 Excluindo as alterações de propriedade padrão do disparador de propriedade IsMouseOver = True.

7) Edite o disparador de propriedade **IsPressed = True**. **78**

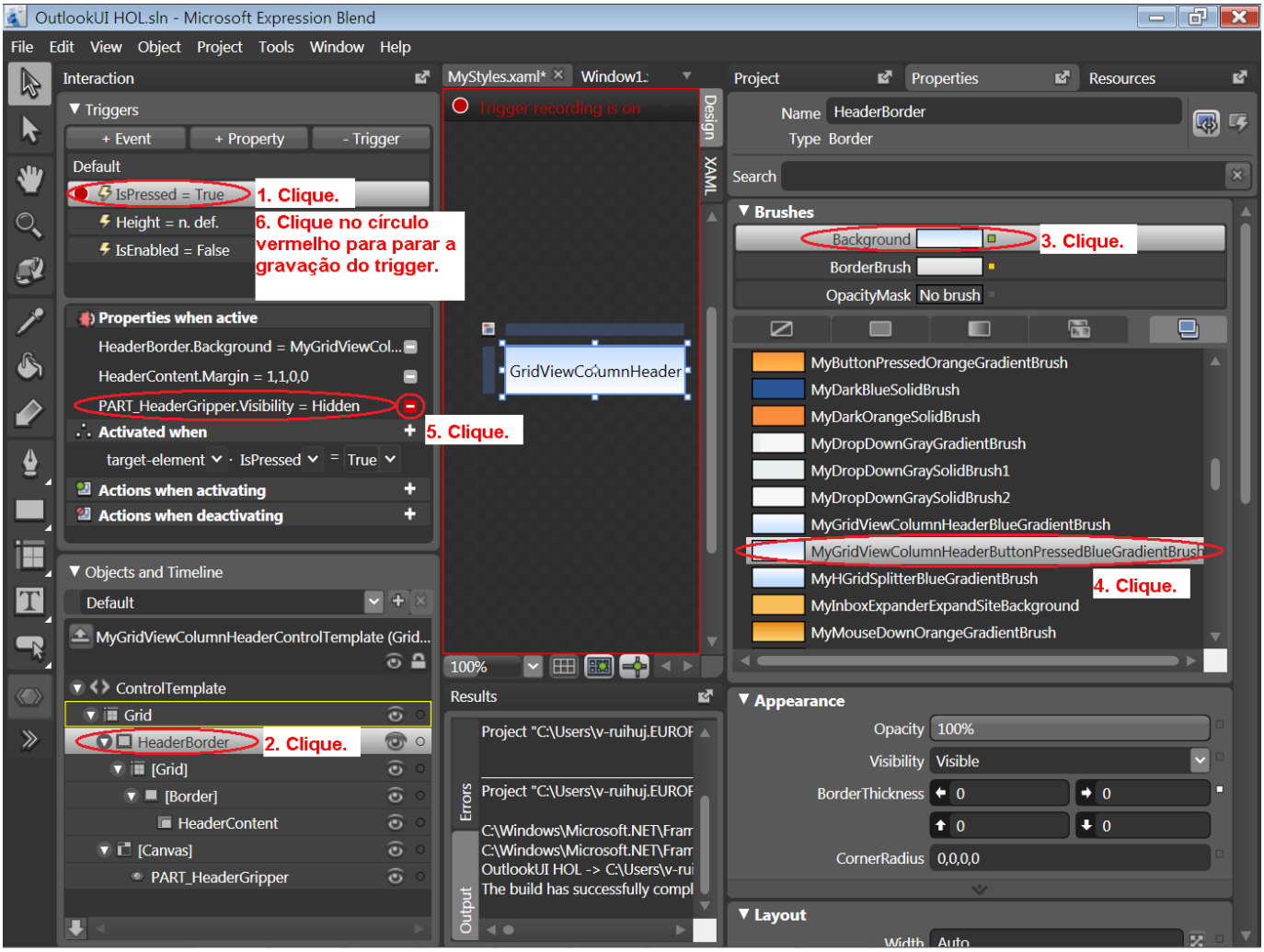


Figura 78 Editando o disparador de propriedade IsPressed = True.

8) Em **Objects and Timeline**, clique em (espaço acima) para sair do **Template Editor** para o **MyGridViewColumnHeaderControlTemplate**.

Crie e execute o aplicativo (**F5**), verifique se os cabeçalhos de coluna **From**, **To** e **Subject** têm a mesma aparência que o MS Outlook 2007.

Tarefa 2: Editar o ItemContainerStyle do [ListView]

Nesta tarefa, vamos configurar o **ItemContainerStyle** do **[ListView]** para personalizar a aparência de um item selecionado da lista e de itens desmarcados da lista. Além disso, como no MS Outlook, faremos com que os emails não lidos sejam exibidos em negrito.

1. Entre no **Style Editor** no **ItemContainerStyle** do **[ListView]**.
   Com **[ListView]** selecionado, abra o menu **Object**, selecione **Edit Other Styles > Edit ItemContainerStyle > Create Empty…**. Na caixa de diálogo, nomeie o estilo **MyListViewItemContainerStyle** e escolha **MyStyles.xaml** como resource dictionary.
2. Configure as propriedades da seguinte maneira:
   Layout
   - **HorizontalContentAlignment = Stretch**
   - **Padding: left = 0,** right = 0, **top = 1.5, bottom = 1.5**

   Text
   - **FontSize = 11**
3. Edite o modelo de controle do **MyListViewItemContainerStyle**.
   1) Entre no editor de modelo de controle do **MyListViewItemContainerControlTemplate**.
      Em **Objects and Timeline**, clique com o botão direito do mouse em **Style** para abrir o menu contextual, escolha **Edit Control Parts (Template) > Edit a Copy…**. Na caixa de diálogo, nomeie o modelo de controle **MyListViewItemContainerControlTemplate** e escolha **MyStyles.xaml** como resource dictionary.
   2) Exclua o **[ContentPresenter]**.
   3) Com **Bd** ativado, adicione **GridViewRowPresenter** no **Bd**.
   4) Configure as propriedades do **Bd** da seguinte maneira:
      Brushes
      - **BorderBrush = MyBlueSolidBrush1**

      Appearance
      - **BorderThickness:** left = 0, right = 0, top = 0, **bottom = 0.5**
   5) Exclua as alterações da propriedade padrão do disparador de propriedade **IsSelected = True** e adicione as seguintes alterações de propriedades para o disparador de propriedade **IsSelected = True**:
      Brushes
      - **Bd.Background = MySelectedListItemBlueSolidBrush**
      - **Bd.BorderBrush = Black** (Insira **Black** na caixa de diálogo **Custom Expression**)

      Appearance
      - **Bd.BorderThickness = 0.5, 0.5, 0.5, 0.5** (Insira **0.5** na caixa de diálogo **Custom Expression**)

      Layout
      - **Bd.Margin:** left = 0, right = 0, **top = -1,** bottom = 0
   6) Excluir as alterações da propriedade padrão do disparador de propriedade **IsSelected = True e IsSelectionActive = False**, e adicione as seguintes alterações de propriedade no disparador de propriedade:
      Brushes
      - **Bd.Background = MyGraySolidBrush**
      - **Bd.BorderBrush = MyBlueSolidBrush1**
   7) Saia do editor de modelo de controle.
4. Faça os emails não lidos serem exibidos em negrito.
   Com **MyStyles.xaml** sendo o arquivo editável atual, alterne para o **View XAML**. Localize **MyListViewItemContainerStyle** (abra a caixa de diálogo **Find** pressionando **CTRL+F**, em seguida, digite **MyListViewItemContainerStyle** na caixa de texto e clique em **Find Next**) e adicione o seguinte texto vermelho no **MyListViewItemContainerStyle**:

listview2.txt

```
<Style x:Key="MyListViewItemContainerStyle" TargetType="{x:Type ListViewItem}">
    <Setter Property="HorizontalContentAlignment" Value="Stretch"/>
    <Setter Property="Padding" Value="0,1.5,0,1.5"/>
    <Setter Property="FontSize" Value="11"/>
    <Setter Property="Template" Value="{DynamicResource
MyListViewItemContainerControlTemplate}"/>
    <Style.Triggers>
        <DataTrigger Binding="{Binding XPath=Read}" Value="false">
            <Setter Property="TextBlock.FontWeight" Value="Bold"/>
        </DataTrigger>
    </Style.Triggers>
</Style>
```

DataTrigger

Style, **ControlTemplate** e **DataTemplate** todos têm uma coleção de disparadores. Um **DataTrigger** permite que você defina valores de propriedade quando o valor de propriedade de um objeto de dados coincide com um valor especificado. No nosso exemplo, é definida a espessura da fonte de um item de lista para negrito se a propriedade de Leitura do email correspondente tem o valor falso.
Note que você deve especificar as propriedades **Binding** e **Value** em um **DataTrigger** para que o disparador de dados seja significativo. Se uma ou ambas as propriedades não forem especificadas, será apresentada uma exceção. Os **Setter**s são usados para especificar os valores de propriedade que tenham efeito quando o objeto de dados relacionado estiver conforme a condição especificada.

Tarefa 3: Adicionar Colunas de Imagem no Importance, Read, Attachment do [ListView]

A primeira coluna do **[ListView]** deve exibir ! se o email for marcado como importante. Para fazer isso, precisamos apresentar outro importante conceito no WPF: **Data Templating**.

Data Templating

DataTemplate (modelo de dados) é sobre a apresentação e aparência dos objetos de dados e é um dos muitos recursos fornecidos pelo modelo de template e estilo WPF. O que você especificar no seu **DataTemplate** se torna a estrutura visual do seu objeto de dados. Os controles WPF têm funcionalidades internas para suportar a personalização da apresentação de dados.

1. Crie um modelo de dados **MyImportanceCellDataTemplate**.

   Nosso modelo de dados para as células na coluna **Importance** tem um controle **Image** chamado **myImage**, ele é usado para mostrar se o email está marcado como importante ou não. O controle **Image myImage** não mostra nenhuma imagem se o elemento **Importance** do **mail** relacionado tiver o valor **false**; e ele mostra **"graphics\important.gif"** se **Importance** for igual a **true**. Isso significa que o valor de **myImage.Source** depende do valor de **Importance** do **mail**, que nós disparamos, para usar um disparador de dados, para definir valores de propriedade diferentes para casos diferentes.
   Alterne para o **View XAML** do **[ListView]** e adicione o seguinte texto vermelho no **[ListView]** (como o modelo de dados só é usado no **[ListView]**, faz sentido definir o modelo de dados localmente):

   listview3.txt

```
<ListView … >
    <ListView.Resources>
        <DataTemplate x:Key="MyImportanceCellDataTemplate">
            <Image x:Name="myImage" Stretch="None" />
            <DataTemplate.Triggers>
                <DataTrigger Binding="{Binding XPath=Importance}" Value="true" >
                    <Setter Property="Source" TargetName="myImage"
                                        Value="graphics\important.gif" />
                </DataTrigger>
            </DataTemplate.Triggers>
        </DataTemplate>
    </ListView.Resources>
    <ListView.View>…</ListView.View>
</ListView>
```

2. Adicione o seguinte texto vermelho no **[ListView]**, no **Window1.xaml**, para criar uma coluna de imagem:

   listview3.txt

```
<ListView … >
    …
    <ListView.View>
        <GridView>
            <GridViewColumn CellTemplate="{DynamicResource
                                        MyImportanceCellDataTemplate}" >
                <GridViewColumn.Header>
                    <Image Source="graphics\importance.gif" Stretch="None" />
                </GridViewColumn.Header>
            </GridViewColumn>
            …
        </GridView>
    </ListView.View>
</ListView>
```

A segunda coluna do **[ListView]** mostra se o email foi lido (**Read = true**) e se o email não foi lido pelo usuário (**Read = true**).

1. Adicione o seguinte texto no **ListView.Resources** do **[ListView]**:

listview3.txt

```
<DataTemplate x:Key="MyReadCellDataTemplate">
    <Image x:Name="myImage" Stretch="None" />
    <DataTemplate.Triggers>
        <DataTrigger Binding="{Binding XPath=Read}" Value="true" >
            <Setter Property="Source" TargetName="myImage"
                                        Value="graphics\read.gif" />
        </DataTrigger>
        <DataTrigger Binding="{Binding XPath=Read}" Value="false" >
            <Setter Property="Source" TargetName="myImage"
                                        Value="graphics\unread.gif" />
        </DataTrigger>
    </DataTemplate.Triggers>
</DataTemplate>
```

2. Adicione o seguinte texto vermelho no **[ListView]**, no **Window1.xaml**, para criar uma segunda coluna de imagem:

listview3.txt

```
<ListView … >
    <ListView.View>
        <GridView>
            …
            <GridViewColumn CellTemplate="{DynamicResource MyReadCellDataTemplate}">
                <GridViewColumn.Header>
                    <Image Source="graphics\mailType.gif" Stretch="None" />
                </GridViewColumn.Header>
            </GridViewColumn>
            <GridViewColumn Header="From" … />
        </GridView>
    </ListView.View>
</ListView>
```

A terceira coluna do **[ListView]** é também uma coluna de imagem, ela é usada para mostrar se o email têm um anexo enviado com ele. Se sim (**mail.Attachment=file name**), então é exibido, caso contrário (**mail.Attachment=-1**), nenhuma imagem é mostrada.

1. Adicione o seguinte texto no **ListView.Resources** do **[ListView]**:

listview3.txt

```
<DataTemplate x:Key="MyAttachmentCellDataTemplate">
    <Image x:Name="myImage" Source="graphics\attachment.gif" Stretch="None" />
    <DataTemplate.Triggers>
        <DataTrigger Binding="{Binding XPath=Attachment}" Value="-1" >
            <Setter Property="Source" TargetName="myImage" Value="{x:Null}" />
        </DataTrigger>
    </DataTemplate.Triggers>
</DataTemplate>
```

2. Adicione o seguinte texto vermelho no **[ListView]**, no **Window1.xaml**, para criar uma terceira coluna de imagem:

listview3.txt

```
<ListView … >
    <ListView.View>
        <GridView>
            …
            <GridViewColumn CellTemplate="{DynamicResource
                                        MyAttachmentCellDataTemplate}">
                <GridViewColumn.Header>
                    <Image Source="graphics\attachment.gif" Stretch="None" />
                </GridViewColumn.Header>
            </GridViewColumn>
            <GridViewColumn Header="From" … />
            …
        </GridView>
    </ListView.View>
</ListView>
```

Tarefa 4: Classificar a lista

Uma lista sem a funcionalidade de classificação dificilmente é uma lista útil, portanto, a próxima etapa é adicionar a funcionalidade de classificação para permitir que o usuário classifique a lista por **Importance**, **Read**, **Attachment**, **Sender**, **Receiver** e **Subject**.

A lista é classificada quando o usuário clica em um determinado cabeçalho da coluna. Dependendo da ordem de classificação (crescente ou decrescente) adicionamos uma seta para cima ou para baixo no cabeçalho da coluna, como mostrado na figura 79:

Figura 79 Cabeçalho da coluna com a seta para cima ou para baixo.

As diferentes apresentações dos cabeçalhos da coluna são novamente obtidas usando modelos de dados.

1. Adicione os seguintes modelos de dados no **ListView.Resources** do **[ListView]**:

listview4.txt

```
<DataTemplate x:Key="MyArrowUpColumnHeaderTemplate">
    <StackPanel Orientation="Horizontal" >
        <TextBlock Text="{Binding}" />
        <Path StrokeThickness="1" Fill="{DynamicResource MyBlueSolidBrush2}"
                                    Data="M 0,4 L 3.5,0 L 7,4 Z" Margin="10,7,0,0"/>
    </StackPanel>
</DataTemplate>

<DataTemplate x:Key="MyArrowDownColumnHeaderTemplate">
    <StackPanel Orientation="Horizontal" >
        <TextBlock Text="{Binding}" />
        <Path StrokeThickness="1" Fill="{DynamicResource MyBlueSolidBrush2}"
                                    Data="M 0,0 L 3.5,4 L 7,0 Z" Margin="10,7,0,0"/>
    </StackPanel>
</DataTemplate>
```

2. Adicione o seguinte texto vermelho ao elemento **[ListView]**, no **Window1.xaml**, para registrar o manipulador de eventos **myGridViewColumnHeader_Click**:

listview4.txt

```
<ListView ButtonBase.Click="myGridViewColumnHeader_Click" x:Name="myListView" … >
```

3. Adicione o seguinte código à classe **Window1** no **Window1.xaml.cs** para a funcionalidade de classificação do **myListView**:

listview4.txt

```
private GridViewColumnHeader lastHeaderClicked = null;
private ListSortDirection lastDirection = ListSortDirection.Ascending;

private void myGridViewColumnHeader_Click(object sender, RoutedEventArgs e)
{
  GridViewColumnHeader headerClicked = e.OriginalSource as GridViewColumnHeader;
  ListSortDirection direction;

  if (headerClicked != null) {
    if (headerClicked.Role != GridViewColumnHeaderRole.Padding) {
      // determine the sort order
      if (headerClicked != lastHeaderClicked) {
          direction = ListSortDirection.Ascending;
      } else {
```

```
            if (lastDirection == ListSortDirection.Ascending) {
                direction = ListSortDirection.Descending;
            } else {
                direction = ListSortDirection.Ascending;
            }
        }

        // sort the list
        if (headerClicked.Column.Header is string) {
            string header = headerClicked.Column.Header as string;
            Sort(header, direction);
        } else if (headerClicked.Column.Header is Image) {
            Image header = headerClicked.Column.Header as Image;
            string type = header.Source.ToString();
            if (type.Contains("attachment")) {
                Sort("Attachment", direction);
            } else if (type.Contains("mailType")) {
                Sort("Read", direction);
            } else if (type.Contains("importance")) {
                Sort("Importance", direction);
            } else {
                return;
            }
        } else {
            return;
        }

        // Add arrow to the column header if it is a string
        if (headerClicked.Column.Header is string) {
            if (direction == ListSortDirection.Ascending) {
                headerClicked.Column.HeaderTemplate =
                        ((ListView)sender).Resources["MyArrowUpColumnHeaderTemplate"] as DataTemplate;
            } else {
                headerClicked.Column.HeaderTemplate =
                        ((ListView)sender).Resources["MyArrowDownColumnHeaderTemplate"] as DataTemplate;
            }
        }
        // Remove arrow from previously sorted header
        if (lastHeaderClicked != null && lastHeaderClicked != headerClicked) {
            lastHeaderClicked.Column.HeaderTemplate = null;
        }

        // Update sorting information
        lastHeaderClicked = headerClicked;
        lastDirection = direction;
    }
  }
}

private void Sort(string sortBy, ListSortDirection direction)
{
  ICollectionView dataView =
      CollectionViewSource.GetDefaultView(myListView.ItemsSource);
  dataView.SortDescriptions.Clear();
  SortDescription sd = new SortDescription(sortBy, direction);
  dataView.SortDescriptions.Add(sd);
}
```

É realmente admirável que você tenha permanecido no curso e terminado este Laboratório Prático. Esperamos que você tenha apreciado o laboratório. WPF é uma estrutura muito boa para desenvolver poderosos aplicativos Windows. Se você agora tem a mesma opinião que nós temos, então nosso trabalho foi bem feito, e nós estaremos contentes se você considerasse o WPF como uma opção quando você estiver criando seus projetos. Obrigado!

Mais em Expression Blend e Windows Presentation Foundation

O Microsoft Expression Blend é fornecido com um guia do usuário detalhado. Para ler o guia do usuário, abra o menu **Help** e escolha, então, **User Guide**.

O web site do Microsoft Expression Blend é
http://www.microsoft.com/Expression/products/overview.aspx?key=blend

Para mais treinamentos do Expression Blend, vá para
http://www.microsoft.com/Expression/kc/product.aspx?key=blend

Windows Presentation Foundation no MSDN:
http://msdn2.microsoft.com/en-us/library/ms754130.aspx